मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान

भोपाल - 462003 (म.प्र.)

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL -462003 (M. P.)

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

प्रयोगशाला निर्देशिका

LAB MANUAL

सीएसई- द्वितीय वर्ष / तृतीय सेमेस्टर (CSE- II Year / III Semester)
सत्र: जुलाई-दिसंबर, २०२३ (Session: July-Dec, 2023)

डेटा संरचनाएं प्रयोगशाला (सीएसई-२१५)

Data Structures Lab (CSE-215)

विषय समन्वयक (Subject Coordinators)

डॉ. मीनू चावला (Dr. Meenu Chawala)
डॉ. विजय भास्कर (Dr. Vijay Bhaskar)
डॉ. वैभव सोनी (Dr. Vaibhav Soni)
डॉ. राजेश वाधवानी (Dr. Rajesh Wadhvani)
डॉ. जयत्रिलोक चौधरी (Dr. Jaytrilok Chowdary)
प्रो.ममता स्वर्णकार (Prof. Mamta Swarnkar)

शिक्षण सहायक (Teacher Assistants)

अंकिता चौधरी (Ankita Choudhary) (पीएचडी ५ / PhD 5)
ऋचा गोयनका (Richa Goenka) (पीएचडी १३ / PhD 13)
पुनीत हिमथानी (Puneet Himthani) (पीएचडी ३३ / PhD 33)
प्रतिभा टोकस (Pratibha Tokas) (पीएचडी २५ / PhD 25)
स्वप्निल मुराई (Swapnil Murai) (पीएचडी ३४ / PhD 34)
साधना बिजरोठिया (Sadhna Bijrothiya) (पीएचडी ११ / PhD 11)
मिथुन सिंह अहिरवार (Mithun Singh Ahirwar) (पीएचडी १८ / PhD 18)
हिमांशु यादव (Himanshu Yadav) (पीएचडी ३५ / PhD 35)
चेतना इंदौरकर(Chetna Indaurkar) (पीएचडी ३२ / PhD 32)

प्रयोग-१/ Experiment-1

ऐरे (Arrays)

1. किसी दिए गए ऐरे में न्यूनतम और अधिकतम एलिमेंट खोजें ।
   Write a program to find the minimum and maximum element in a given array.

समाधान/Solution:

```
#include <stdio.h>

    int main()
    {
       int size;
       printf("Enter the size of the array: ");
       scanf("%d", &size);

       int arr[size];
       printf("Enter the elements of the array:\n");
       for (int i = 0; i < size; i++)
        {
          scanf("%d", &arr[i]);
        }

       int min = arr[0];
       int max = arr[0];

       for (int i = 1; i < size; i++)
       {
          if (arr[i] < min)
            {
              min = arr[i];
            }
          if (arr[i] > max)
            {
              max = arr[i];
            }
        }

       printf("Array elements: ");
       for (int i = 0; i < size; i++)
       {
```

```
            printf("%d ", arr[i]);
        }
        printf("\n");

        printf("Minimum element: %d\n", min);
        printf("Maximum element: %d\n", max);

        return 0;
}
```

आउटपुट/Output:

Enter the size of the array: 5

Enter the elements of the array:

23 56 12 8 45

Array elements: 23 56 12 8 45

Minimum element: 8

Maximum element: 56

2. अवर्गीकृत इन्टीजर्स के एक ऐरे में मध्यांक ढूंढें ।

Find the median in the unsorted array of integers.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>
int compare(const void *a, const void *b)
{
   return (*(int *)a - *(int *)b);
}

double findMedian(int arr[], int size)
{
   qsort(arr, size, sizeof(int), compare);

   if (size % 2 == 0)
         {
      int mid1 = arr[size / 2 - 1];
      int mid2 = arr[size / 2];
      return (double)(mid1 + mid2) / 2.0;
         }
else
         {
      return (double)arr[size / 2];
         }
}

int main()
 {
   int size;
   printf("Enter the size of the array: ");
   scanf("%d", &size);
   int arr[size];
   printf("Enter the elements of the array:\n");
   for (int i = 0; i < size; i++)
         {
      scanf("%d", &arr[i]);
         }
double median = findMedian(arr, size);
printf("Array elements: ");
    for (int i = 0; i < size; i++)
         {
```

```
        printf("%d ", arr[i]);
          }
    printf("\n");
    printf("Median: %.2f\n", median);

    return 0;
}
```

आउटपुट/Output:

```
Enter the size of the array: 5
Enter the elements of the array:
23 56 12 8 45
Array elements: 23 56 12 8 45
Median: 23.00
```

3. अवर्गीकृत ऐरे में दूसरे सबसे बड़े एलिमेंट का पता लगाएँ।
Find the second largest element in an unsorted array.

समाधान/Solution:

```
#include <stdio.h>
void findSecondLargest(int arr[], int size)
{
   int largest = arr[0];
   int secondLargest = arr[0];
 for (int i = 1; i < size; i++)
{
       if (arr[i] > largest)
          {
          secondLargest = largest;
          largest = arr[i];
          }
else if (arr[i] > secondLargest && arr[i] != largest)
         {
          secondLargest = arr[i];
          }
   }
 if (secondLargest != largest)
         {
       printf("Second largest element: %d\n", secondLargest);
          }
else
         {
       printf("There is no second largest element.\n");
          }
}
```

```
int main()
{
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++)
            {
        scanf("%d", &arr[i]);
            }
  findSecondLargest(arr, size);
 return 0;
}
```

आउटपुट/Output:

Enter the size of the array: 7

Enter the elements of the array:

23 56 12 8 45 67 34

Second largest element: 56

प्रयोग-२/ Experiment-2

ऐरे (Arrays)

4. दो अवर्गीकृत ऐरे के यूनियन और इंटरसेक्शन को खोजने का प्रोग्राम लिखिए।
Write a program for finding the union and intersection of two unsorted arrays.

समाधान/Solution:

```
#include <stdio.h>
void findUnion(int arr1[], int size1, int arr2[], int size2)
{
    int unionArr[size1 + size2];
    int unionSize = 0;
    for (int i = 0; i < size1; i++)
          {
       unionArr[unionSize++] = arr1[i];
          }
     for (int i = 0; i < size2; i++)
          {
       int found = 0;
       for (int j = 0; j < size1; j++)
          {
          if (arr2[i] == arr1[j])
                  {
             found = 1;
             break;
                   }
          }
       if (!found)
          {
```

```
            unionArr[unionSize++] = arr2[i];
            }
    }
 printf("Union of arrays: ");
    for (int i = 0; i < unionSize; i++)
          {
        printf("%d ", unionArr[i]);
          }
    printf("\n");
}
void findIntersection(int arr1[], int size1, int arr2[], int size2)
{
    printf("Intersection of arrays: ");
    for (int i = 0; i < size1; i++)
          {
        for (int j = 0; j < size2; j++)
          {
           if (arr1[i] == arr2[j])
                  {
              printf("%d ", arr1[i]);
              break;
                  }
          }
    }
    printf("\n");
}
int main()
{
    int size1, size2;
```

```
    printf("Enter the size of the first array: ");
    scanf("%d", &size1);
    int arr1[size1];
    printf("Enter the elements of the first array:\n");
    for (int i = 0; i < size1; i++)
          {
       scanf("%d", &arr1[i]);
          }
    printf("Enter the size of the second array: ");
    scanf("%d", &size2);
     int arr2[size2];
    printf("Enter the elements of the second array:\n");
    for (int i = 0; i < size2; i++)
          {
       scanf("%d", &arr2[i]);
           }
    findUnion(arr1, size1, arr2, size2);
    findIntersection(arr1, size1, arr2, size2);
    return 0;
}
```

आउटपुट/Output:

```
Enter the size of the first array: 5
Enter the elements of the first array:   4 7 2 9 1
Enter the size of the second array: 4
Enter the elements of the second array:  7 5 2 6
Union of arrays:   4 7 2 9 1 5 6
Intersection of arrays: 7 2
```

5. दो 2D-आव्यूह को गुणा करने का प्रोग्राम लिखिए ।
Write a program for multiplying two 2D-matrices.

समाधान/Solution:

```
#include <stdio.h>

#define MAX_ROWS 100
#define MAX_COLS 100

void multiplyMatrices(int mat1[][MAX_COLS], int rows1, int cols1,
                      int mat2[][MAX_COLS], int rows2, int cols2,
                      int result[][MAX_COLS])
{
    if (cols1 != rows2)
          {
        printf("Matrix multiplication is not possible.\n");
        return;
          }

    for (int i = 0; i < rows1; i++)
          {
           for (int j = 0; j < cols2; j++)
                    {
           result[i][j] = 0;
                     for (int k = 0; k < cols1; k++)
                              {
              result[i][j] += mat1[i][k] * mat2[k][j];
                              }
                    }
           }
}

void displayMatrix(int mat[][MAX_COLS], int rows, int cols)
{
    for (int i = 0; i < rows; i++)
          {
        for (int j = 0; j < cols; j++)
          {
           printf("%d ", mat[i][j]);
          }
```

```
        printf("\n");
    }
}

int main()
{
    int mat1[MAX_ROWS][MAX_COLS], mat2[MAX_ROWS][MAX_COLS],
result[MAX_ROWS][MAX_COLS];
    int rows1, cols1, rows2, cols2;

    printf("Enter the number of rows and columns for the first matrix: ");
    scanf("%d %d", &rows1, &cols1);

    printf("Enter the elements of the first matrix:\n");
    for (int i = 0; i < rows1; i++)
          {
         for (int j = 0; j < cols1; j++)
                  {
          scanf("%d", &mat1[i][j]);
                  }
         }

    printf("Enter the number of rows and columns for the second matrix: ");
    scanf("%d %d", &rows2, &cols2);

    printf("Enter the elements of the second matrix:\n");
    for (int i = 0; i < rows2; i++)
         {
       for (int j = 0; j < cols2; j++)
                  {
          scanf("%d", &mat2[i][j]);
                   }
         }

    if (cols1 != rows2)
         {
       printf("Matrix multiplication is not possible.\n");
       return 1;
          }
multiplyMatrices(mat1, rows1, cols1, mat2, rows2, cols2, result);
printf("Resultant matrix after multiplication:\n");
displayMatrix(result, rows1, cols2);
return 0;
```

```
}
```

आउटपुट/Output:

```
Enter the number of rows and columns for the first matrix: 2 3
Enter the elements of the first matrix:
2 3 4
5 6 7
Enter the number of rows and columns for the second matrix: 3 2
Enter the elements of the second matrix:
8 9
10 11
12 13
Resultant matrix after multiplication:
53 58
134 148
```

प्रयोग-३ / Experiment-3

रिकर्शन (Recursion)

1. रिकर्शन का उपयोग करके nवीं फिबोनाकी संख्या ज्ञात करें ।
   Find the nth fibonacci number using recursion.

समाधान/Solution:

```
#include <stdio.h>
int fibonacci(int n)
{
    if (n <= 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}
int main()
{
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    int result = fibonacci(n);
    printf("The %dth Fibonacci number is: %d\n", n, result);
    return 0;
}
```

आउटपुट/Output:

Enter the value of n: 10

The 10th Fibonacci number is: 55

2. रिकर्शन का उपयोग करके 2 संख्याओं का महत्तम समापवर्तक ज्ञात करें ।
Find the greatest common divisor of 2 numbers using recursion.

समाधान/Solution:

```
#include <stdio.h>

int gcd_recursive(int a, int b)
{
    if (b == 0)
{
        return a;
    } else {
        return gcd_recursive(b, a % b);
    }
}

int main() {
    int num1 = 48;
    int num2 = 18;
    int result = gcd_recursive(num1, num2);
    printf("The GCD of %d and %d is %d\n", num1, num2, result);
    return 0;
}
```

आउटपुट/Output:

The GCD of 48 and 18 is 6

3.रिकर्शन का उपयोग करके इन्टीजर ऐरे के सभी एलिमेंट का योग ज्ञात करें ।

Find the sum of all elements of an integer array using recursion.

समाधान/Solution:

```
#include <stdio.h>

int sum_recursive(int arr[], int n) {
    if (n <= 0) {
        return 0;
    } else {
        return arr[n - 1] + sum_recursive(arr, n - 1);
    }
}

int main() {
    int arr[] = {5, 10, 15, 20, 25};
    int size = sizeof(arr) / sizeof(arr[0]);
    int result = sum_recursive(arr, size);
    printf("Sum of array elements: %d\n", result);
    return 0;
}
```

आउटपुट/Output:

```
Sum of array elements: 75
```

4.'n' डिस्क वाले टावर्स ऑफ हनोई की समस्या का समाधान खोजने के लिए प्रोग्राम लिखिए ।

Write a program for finding the solution of towers of hanoi problem having 'n' discs.

समाधान/Solution:

```
#include <stdio.h>

void towerOfHanoi(int n, char source, char auxiliary, char destination) {
    if (n == 1) {
        printf("Move disk 1 from %c to %c\n", source, destination);
        return;
    }

    towerOfHanoi(n - 1, source, destination, auxiliary);
    printf("Move disk %d from %c to %c\n", n, source, destination);
    towerOfHanoi(n - 1, auxiliary, source, destination);
}

int main() {
    int n;
    printf("Enter the number of discs: ");
    scanf("%d", &n);

    towerOfHanoi(n, 'A', 'B', 'C');

    return 0;
}
```

आउटपुट/Output:

```
Enter the number of discs: 3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

प्रयोग-४ / Experiment-4

लिंक्ड लिस्ट (Linked List)

1.लिंक्ड लिस्ट में मध्य एलिमेंट खोजने के लिए प्रोग्राम लिखिए ।

Write a program for finding the middle element in the linked list.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node* next;
};

struct Node* createNode(int data)
 {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void findMiddle(struct Node* head)
{
    struct Node* slowPtr = head;
    struct Node* fastPtr = head;

    if (head != NULL) {
        while (fastPtr != NULL && fastPtr->next != NULL)
        {
            fastPtr = fastPtr->next->next;
            slowPtr = slowPtr->next;
        }
        printf("The middle element is: %d\n", slowPtr->data);
    }
}

int main()
 {
    struct Node* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);
```

```
    head->next->next->next = createNode(4);
    head->next->next->next->next = createNode(5);

    findMiddle(head);

    return 0;
}
```

आउटपुट/Output:

The middle element is: 3

2.लिंक्ड लिस्ट को रिवर्सिंग के लिए प्रोग्राम लिखिए ।

Write a program for reversing the linked list.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void printList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

struct Node* reverseList(struct Node* head) {
    struct Node* prev = NULL;
    struct Node* current = head;
    struct Node* next = NULL;

    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }

    return prev;
}

int main() {
    struct Node* head = createNode(1);
    head->next = createNode(2);
```

```
    head->next->next = createNode(3);
    head->next->next->next = createNode(4);
    head->next->next->next->next = createNode(5);

    printf("Original linked list: ");
    printList(head);

    head = reverseList(head);

    printf("Reversed linked list: ");
    printList(head);

    return 0;
}
```

आउटपुट/Output:

```
Original linked list: 1 2 3 4 5
Reversed linked list: 5 4 3 2 1
```

3.लिंक्ड लिस्ट में सम और विषम की-नोड्स को अलग करने का प्रोग्राम लिखिए ।

Write a program for segregating even and odd key nodes in a linked list.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void printList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

struct Node* segregateEvenOdd(struct Node* head) {
    struct Node* evenHead = NULL;
    struct Node* evenTail = NULL;
    struct Node* oddHead = NULL;
    struct Node* oddTail = NULL;
    struct Node* current = head;

    while (current != NULL) {
        if (current->data % 2 == 0) {
            if (evenHead == NULL) {
                evenHead = current;
                evenTail = current;
            } else {
                evenTail->next = current;
                evenTail = evenTail->next;
            }
        }
```

```
else
 {
          if (oddHead == NULL)
       {
            oddHead = current;
            oddTail = current;
          } else {
            oddTail->next = current;
            oddTail = oddTail->next;
          }
       }
       current = current->next;
   }

   if (evenHead == NULL)
       {
       return oddHead;
   } else {
       evenTail->next = oddHead;
       if (oddTail != NULL) {
          oddTail->next = NULL;
       }
       return evenHead;
   }
}

int main()
{
   struct Node* head = createNode(17);
   head->next = createNode(15);
   head->next->next = createNode(8);
   head->next->next->next = createNode(12);
   head->next->next->next->next = createNode(10);

   printf("Original linked list: ");
   printList(head);

   head = segregateEvenOdd(head);

   printf("Segregated linked list: ");
   printList(head);

   return 0;
}
```

आउटपुट/Output:

```
Original linked list: 17 15 8 12 10
Segregated linked list: 8 12 10 17 15
```

4.लिंक्ड लिस्ट में लूप का पता लगाने के लिए प्रोग्राम लिखिए ।

Write a program for detecting a loop in a linked list.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

// Definition of a linked list node
struct Node {
    int data;
    struct Node* next;
};

// Function to insert a new node at the end of the linked list
void insert(struct Node** head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL)
{
        *head = newNode;
    }
          Else
          {
        struct Node* temp = *head;
        while (temp->next != NULL)
          {
           temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Function to detect a loop in a linked list
int detectLoop(struct Node* head)
 {
    struct Node* slow = head;
    struct Node* fast = head;
```

```
    while (fast != NULL && fast->next != NULL)
{
        slow = slow->next;
        fast = fast->next->next;

        if (slow == fast) {
            return 1; // Loop detected
        }
    }

    return 0; // No loop detected
}

int main()
{
    struct Node* head = NULL;

    insert(&head, 1);
    insert(&head, 2);
    insert(&head, 3);
    insert(&head, 4);
    insert(&head, 5);

    // Create a loop for testing (connecting the last node to the second node)
    head->next->next->next->next->next = head->next;

    if (detectLoop(head)) {
        printf("Loop detected in the linked list.\n");
    } else {
        printf("No loop detected in the linked list.\n");
    }

    return 0;
}
```

आउटपुट/Output:

Loop detected in the linked list.

प्रयोग-५ / Experiment-5

लिंक्ड लिस्ट (Linked List)

5.जांचें कि दी गई लिंक्ड लिस्ट पैलिंड्रोम है या नहीं ।
Check whether a given linked list is palindrome or not.

समाधान/Solution:

5, 4,3, 11, 25, 45, 11, 58, 84, 89, 48, 47, 74, 57

6. एक सर्कुलर लिंक्ड लिस्ट बनाएं और डेटा को हमेशा पीछे की ओर डालें ।
Create a circular linked list and always insert data at tail.

समाधान/Solution:

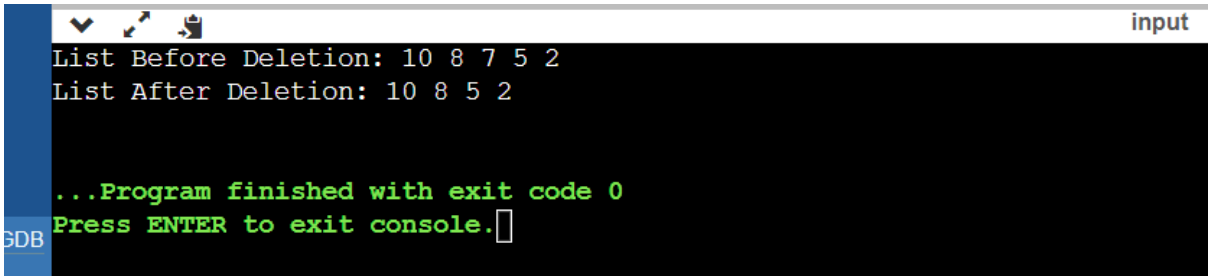
```
#include <stdio.h>
#include <stdlib.h>
// Structure for a node
struct Node {
	int data;
	struct Node* next;
};
// Function to insert a node at the
// beginning of a Circular linked list
void push(struct Node** head_ref, int data)
{
	// Create a new node and make head
	// as next of it.
	struct Node* ptr1 = (struct Node*)malloc(sizeof(struct Node));
	ptr1->data = data;
	ptr1->next = *head_ref;
	// If linked list is not NULL then
	// set the next of last node
	if (*head_ref != NULL) {
		// Find the node before head and
		// update next of it.
		struct Node* temp = *head_ref;
		while (temp->next != *head_ref)
			temp = temp->next;
		temp->next = ptr1;
```

```
        }
        else
                // For the first node
                ptr1->next = ptr1;
        *head_ref = ptr1;
}
// Function to print nodes in a given
// circular linked list
void printList(struct Node* head)
{
        struct Node* temp = head;
        if (head != NULL) {
                do {
                        printf("%d ", temp->data);
                        temp = temp->next;
                } while (temp != head);
        }
        printf("\n");
}
// Function to delete a given node
// from the list
void deleteNode(struct Node** head, int key)
{
        // If linked list is empty
        if (*head == NULL)
                return;
        // If the list contains only a
        // single node
        if ((*head)->data == key && (*head)->next == *head) {
```

```
            free(*head);
            *head = NULL;
            return;
}
struct Node *last = *head, *d;
// If head is to be deleted
if ((*head)->data == key) {
            // Find the last node of the list
            while (last->next != *head)
                        last = last->next;
            // Point last node to the next of
            // head i.e. the second node
            // of the list
            last->next = (*head)->next;
            free(*head);
            *head = last->next;
            return;
}
// Either the node to be deleted is
// not found or the end of list
// is not reached
while (last->next != *head && last->next->data != key) {
            last = last->next;
}
// If node to be deleted was found
if (last->next->data == key) {
            d = last->next;
            last->next = d->next;
            free(d);
```

```
        }
        else
                printf("Given node is not found in the list!!!\n");
}
// Driver code
int main()
{
        // Initialize lists as empty
        struct Node* head = NULL;
        // Created linked list will be
        // 2->5->7->8->10
        push(&head, 2);
        push(&head, 5);
        push(&head, 7);
        push(&head, 8);
        push(&head, 10);
        printf("List Before Deletion: ");
        printList(head);
        deleteNode(&head, 7);
        printf("List After Deletion: ");
        printList(head);
        return 0;
}
```

आउटपुट/Output:



7. दी गई सिंगली लिंक्ड लिस्ट से डबली लिंक्ड लिस्ट बनाएं।
   Create a doubly linked list from a given singly linked list.

समाधान/Solution:

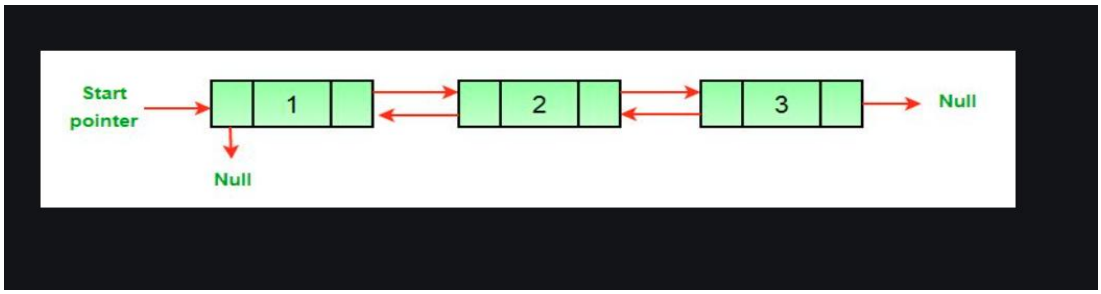
```
#include <stdio.h>
#include <stdlib.h>
struct node {
   int num;
   struct node * preptr;
   struct node * nextptr;
}*stnode, *ennode;
void DlListcreation(int n);
void displayDlList();
int main()
{
   int n;
   stnode = NULL;
   ennode = NULL;
          printf("\n\n Doubly Linked List : Create and display a doubly linked list :\n");
          printf("--------------------------------------------------------------------\n");
   printf(" Input the number of nodes : ");
   scanf("%d", &n);
    DlListcreation(n);
   displayDlList();
   return 0;
}
```

```
 void DlListcreation(int n)
{
   int i, num;
   struct node *fnNode;
    if(n >= 1)
   {
      stnode = (struct node *)malloc(sizeof(struct node));

      if(stnode != NULL)
      {
         printf(" Input data for node 1 : "); // assigning data in the first node
         scanf("%d", &num);
          stnode->num = num;
         stnode->preptr = NULL;
         stnode->nextptr = NULL;
         ennode = stnode;
// putting data for rest of the nodes
         for(i=2; i<=n; i++)
         {
            fnNode = (struct node *)malloc(sizeof(struct node));
            if(fnNode != NULL)
            {
               printf(" Input data for node %d : ", i);
               scanf("%d", &num);
               fnNode->num = num;
               fnNode->preptr = ennode;    // new node is linking with the previous node
               fnNode->nextptr = NULL;
                ennode->nextptr = fnNode;   // previous node is linking with the new node
               ennode = fnNode;           // assign new node as last node
            }
            else
```

```
            {
                printf(" Memory can not be allocated.");
                break;
            }
          }
        }
        else
        {
            printf(" Memory can not be allocated.");
        }
    }
}
void displayDlList()
{
    struct node * tmp;
    int n = 1;
    if(stnode == NULL)
    {
        printf(" No data found in the List yet.");
    }
    else
    {
        tmp = stnode;
        printf("\n\n Data entered on the list are :\n");

        while(tmp != NULL)
        {
            printf(" node %d : %d\n", n, tmp->num);
            n++;
            tmp = tmp->nextptr; // current pointer moves to the next node
        }
```

```
    }
}
```

आउटपुट/Output:



8. डबली लिंक्ड लिस्ट को रिवर्स करने का प्रोग्राम लिखिए ।
Write a program for reversing a doubly linked list.

समाधान/Solution:

```
linked list */
#include <stdio.h>
#include <stdlib.h>
// A node of the doubly linked list
struct Node
{
int data;
struct Node *next;
struct Node *prev;
};

/* Function to reverse a Doubly
Linked List */
void reverse(struct Node **head_ref)
{
    struct Node *temp = NULL;
    struct Node *current = *head_ref;

    /* Swap next and prev for all nodes
        of doubly linked list */
    while (current != NULL)
    {
    temp = current->prev;
    current->prev = current->next;
    current->next = temp;
    current = current->prev;
    }

    /* Before changing head, check for the
        cases like empty list and list with
        only one node */
    if(temp != NULL )
        *head_ref = temp->prev;
}
```

```
// UTILITY FUNCTIONS
/* Function to insert a node at the beginning
of the Doubly Linked List */
void push(struct Node** head_ref,
          int new_data)
{
     // Allocate node
     struct Node* new_node =
          (struct Node*) malloc(sizeof(struct Node));

     // Put in the data
     new_node->data = new_data;
     /* Since we are adding at the beginning,
     prev is always NULL */
     new_node->prev = NULL;

     /* Link the old list of the new node */
     new_node->next = (*head_ref);

     /* Change prev of head node to new node */
     if((*head_ref) != NULL)
     (*head_ref)->prev = new_node ;

     /* move the head to point to the new node */
     (*head_ref) = new_node;
}

/* Function to print nodes in a given
doubly linked list. This function is
same as printList() of singly linked list */
void printList(struct Node *node)
{
while(node!=NULL)
{
printf("%d ", node->data);
node = node->next;
}
}
// Driver code
int main()
{
// Start with the empty list
struct Node* head = NULL;

/* Let us create a sorted linked list
     to test the functions. Created linked list
```

```
      will be 10->8->4->2 */
push(&head, 2);
push(&head, 4);
push(&head, 8);
push(&head, 10);
printf(
"Original Linked list ");
printList(head);

// Reverse doubly linked list
reverse(&head);

printf(
"Reversed Linked list ");
printList(head);
getcha();
}
```

आउटपुट/Output:

प्रयोग-६/ Experiment-6

स्टैकस (Stacks)

1. ऐरे का उपयोग करके बढ़ने योग्य स्टैक का प्रोग्राम लिखिए ।
   Write a program to implement a growable stack using arrays.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

#define INITIAL_CAPACITY 5

struct GrowableStack {
    int *array;
    int top;
    int capacity;
};

void initialize(struct GrowableStack *stack) {
    stack->array = (int *)malloc(INITIAL_CAPACITY * sizeof(int));
    stack->top = -1;
    stack->capacity = INITIAL_CAPACITY;
}

int isEmpty(struct GrowableStack *stack) {
    return stack->top == -1;
```

```
}

void push(struct GrowableStack *stack, int data) {
    if (stack->top == stack->capacity - 1) {
        // If the stack is full, double the capacity
        stack->capacity *= 2;
        stack->array = (int *)realloc(stack->array, stack->capacity * sizeof(int));
    }
    stack->array[++stack->top] = data;
}

int pop(struct GrowableStack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty.\n");
        return -1;
    }
    return stack->array[stack->top--];
}

int main() {
    struct GrowableStack stack;
    initialize(&stack);

    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);

    printf("Popped: %d\n", pop(&stack));
    printf("Popped: %d\n", pop(&stack));
```

```
    printf("Popped: %d\n", pop(&stack));

    push(&stack, 40);
    push(&stack, 50);

    printf("Popped: %d\n", pop(&stack));
    printf("Popped: %d\n", pop(&stack));

    return 0;
}
```

आउटपुट/Output:

```
Popped: 30
Popped: 20
Popped: 10
Popped: 50
Popped: 40
```

2. लिंक्ड लिस्ट की सहायता से स्टैक बनाए ।
   Implement stack with the help of a linked list.

समाधान/Solution:

```
#include <bits/stdc++.h>
using namespace std;
class Node {
public:
	int data;
	Node* link;
	Node(int n)
	{
		this->data = n;
		this->link = NULL;
	}
};
class Stack {
	Node* top;
public:
	Stack() { top = NULL; }
	void push(int data)
	{
		Node* temp = new Node(data);
		if (!temp) {
			cout << "\nStack Overflow";
			exit(1);
		}
		temp->data = data;
		temp->link = top;
		top = temp;
	}
	bool isEmpty()
	{
		return top == NULL;
	}
	int peek()
	{
		if (!isEmpty())
			return top->data;
		else
```

```
                exit(1);
        }
        void pop()
        {
                Node* temp;
                if (top == NULL) {
                        cout << "\nStack Underflow" << endl;
                        exit(1);
                }
                else {
                        temp = top;
                        top = top->link;
                        free(temp);
                }
        }
        void display()
        {
                Node* temp;
                if (top == NULL) {
                        cout << "\nStack Underflow";
                        exit(1);
                }
                else {
                        temp = top;
                        while (temp != NULL) {
                                cout << temp->data;
                                temp = temp->link;
                                if (temp != NULL)
                                        cout << " -> ";
                        }
                }
        }
};
int main()
{
        Stack s;
        s.push(11);
        s.push(22);
        s.push(33);
        s.push(44);
        s.display();
        cout << "\nTop element is " << s.peek() << endl;
        s.pop();
        s.pop();
        s.display();
        cout << "\nTop element is " << s.peek() << endl;
        return 0;
}
```

आउटपुट/Output:

44 -> 33 -> 22 -> 11

Top element is 44

22 -> 11

Top element is 22

3. स्टैक के तत्वों को उलटने के लिए प्रोग्राम लिखिए ।
   Write a program for reversing elements of the stack.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>
#define bool int
struct sNode {
	char data;
	struct sNode* next;
};
void push(struct sNode** top_ref, int new_data);
int pop(struct sNode** top_ref);
bool isEmpty(struct sNode* top);
void print(struct sNode* top);
void insertAtBottom(struct sNode** top_ref, int item)
{
	if (isEmpty(*top_ref))
		push(top_ref, item);
	else {
		int temp = pop(top_ref);
		insertAtBottom(top_ref, item);
		push(top_ref, temp);
	}
}
void reverse(struct sNode** top_ref)
{
	if (!isEmpty(*top_ref)) {
		int temp = pop(top_ref);
		reverse(top_ref);
		insertAtBottom(top_ref, temp);
	}
}
int main()
{
	struct sNode* s = NULL;
	push(&s, 4);
	push(&s, 3);
	push(&s, 2);
	push(&s, 1);
```

```
        printf("\n Original Stack ");
        print(s);
        reverse(&s);
        printf("\n Reversed Stack ");
        print(s);
        return 0;
}
bool isEmpty(struct sNode* top)
{
        return (top == NULL) ? 1 : 0;
}
void push(struct sNode** top_ref, int new_data)
{
        struct sNode* new_node = (struct sNode*)malloc(sizeof(struct sNode));
        if (new_node == NULL) {
                printf("Stack overflow \n");
                exit(0);
        }
        new_node->data = new_data;
        new_node->next = (*top_ref);
        (*top_ref) = new_node;
}
int pop(struct sNode** top_ref)
{
        char res;
        struct sNode* top;
        if (*top_ref == NULL) {
                printf("Stack overflow \n");
                exit(0);
        }
        else {
                top = *top_ref;
                res = top->data;
                *top_ref = top->next;
                free(top);
                return res;
        }
}
void print(struct sNode* top)
{
        printf("\n");
        while (top != NULL) {
                printf(" %d ", top->data);
                top = top->next;
        }
}
```

आउटपुट/Output:

Original Stack

4 3 2 1

Reversed Stack

1 2 3 4

4. स्टैक का उपयोग करके इनफ़िक्स से पोस्टफ़िक्स रूपांतरण के लिए एक प्रोग्राम लिखें।
   Write a program for infix to postfix conversion using stacks.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_EXPR_SIZE 100
int precedence(char operator)
{
	switch (operator) {
	case '+':
	case '-':
		return 1;
	case '*':
	case '/':
		return 2;
	case '^':
		return 3;
	default:
		return -1;
	}
}
int isOperator(char ch)
{
	return (ch == '+' || ch == '-' || ch == '*' || ch == '/'
			|| ch == '^');
}
char* infixToPostfix(char* infix)
{
	int i, j;
	int len = strlen(infix);
	char* postfix = (char*)malloc(sizeof(char) * (len + 2));
	char stack[MAX_EXPR_SIZE];
	int top = -1;
	for (i = 0, j = 0; i < len; i++) {
		if (infix[i] == ' ' || infix[i] == '\t')
			continue;
		if (isalnum(infix[i])) {
```

```
                postfix[j++] = infix[i];
            }
            else if (infix[i] == '(') {
                stack[++top] = infix[i];
            }
            else if (infix[i] == ')') {
                while (top > -1 && stack[top] != '(')
                    postfix[j++] = stack[top--];
                if (top > -1 && stack[top] != '(')
                    return "Invalid Expression";
                else
                    top--;
            }
            else if (isOperator(infix[i])) {
                while (top > -1
                    && precedence(stack[top])
                            >= precedence(infix[i]))
                    postfix[j++] = stack[top--];
                stack[++top] = infix[i];
            }
        }
        while (top > -1) {
            if (stack[top] == '(') {
                return "Invalid Expression";
            }
            postfix[j++] = stack[top--];
        }
        postfix[j] = '\0';
        return postfix;
}
int main()
{
        char infix[MAX_EXPR_SIZE] = "a+b*(c^d-e)^(f+g*h)-i";
        char* postfix = infixToPostfix(infix);
        printf("%s\n", postfix);
        free(postfix);
        return 0;
}
```

आउटपुट/Output:

```
abcd^e-fgh*+^*+i-
```

प्रयोग-७/ Experiment-7

स्टैकस (Stacks)

5. दो क्यू का उपयोग करके स्टैक बनाए ।
   Implement stack using two queues.

समाधान/Solution:

```
#include <bits/stdc++.h>
using namespace std;
class Stack {
	queue<int> q1, q2;
public:
	void push(int x)
	{
		q2.push(x);
		while (!q1.empty()) {
			q2.push(q1.front());
			q1.pop();
		}
		queue<int> q = q1;
		q1 = q2;
		q2 = q;
	}
	void pop()
	{
		if (q1.empty())
			return;
		q1.pop();
	}
	int top()
	{
		if (q1.empty())
			return -1;
		return q1.front();
	}
```

```
    int size() { return q1.size(); }
};

int main()
{
    Stack s;
    s.push(1);
    s.push(2);
    s.push(3);
    cout << "current size: " << s.size() << endl;
    cout << s.top() << endl;
    s.pop();
    cout << s.top() << endl;
    s.pop();
    cout << s.top() << endl;
    cout << "current size: " << s.size() << endl;
    return 0;
}
```

आउटपुट/Output:

```
current size: 3
3
2
1
current size: 1
```

6. पोस्टफ़िक्स फॉर्म में दिए गए समीकरण के मूल्यांकन के लिए स्टैक का उपयोग करें ।
   Use stack for evaluation of equation given in postfix form.

समाधान/Solution:

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Stack {
	int top;
	unsigned capacity;
	int* array;
};
struct Stack* createStack(unsigned capacity)
{
	struct Stack* stack	= (struct Stack*)malloc(sizeof(struct Stack));
	if (!stack)
		return NULL;
	stack->top = -1;
	stack->capacity = capacity;
	stack->array = (int*)malloc(stack->capacity * sizeof(int));
	if (!stack->array)
		return NULL;
	return stack;
}
int isEmpty(struct Stack* stack)
{
	return stack->top == -1;
}
char peek(struct Stack* stack)
{
	return stack->array[stack->top];
}
char pop(struct Stack* stack)
{
	if (!isEmpty(stack))
		return stack->array[stack->top--];
	return '$';
```

```
}
void push(struct Stack* stack, char op)
{
	stack->array[++stack->top] = op;
}
int evaluatePostfix(char* exp)
{
	struct Stack* stack = createStack(strlen(exp));
	int i;
	if (!stack)
		return -1;
	for (i = 0; exp[i]; ++i) {
		if (isdigit(exp[i]))
			push(stack, exp[i] - '0');
		else {
			int val1 = pop(stack);
			int val2 = pop(stack);
			switch (exp[i]) {
			case '+':
				push(stack, val2 + val1);
				break;
			case '-':
				push(stack, val2 - val1);
				break;
			case '*':
				push(stack, val2 * val1);
				break;
			case '/':
				push(stack, val2 / val1);
				break;
			}
		}
	}
	return pop(stack);
}
int main()
{
	char exp[] = "231*+9-";
	printf("postfix evaluation: %d", evaluatePostfix(exp));
	return 0;
}
```

आउटपुट/Output:

```
postfix evaluation: -4
```

7. दो स्टैक समान हैं या नहीं, यह जांचने के लिए एक प्रोग्राम लिखें।
   Write a program to check if two stacks are identical or not.

समाधान/Solution:

```
#include <bits/stdc++.h>
using namespace std;
bool isSameStack(stack<string> stack1, stack<string> stack2)
{
	bool flag = true;
	if (stack1.size() != stack2.size()) {
		flag = false;
		return flag;
	}
	while (stack1.empty() == false) {
		if (stack1.top() == stack2.top()) {
			stack1.pop();
			stack2.pop();
		}
		else {
			flag = false;
			break;
		}
	}
	return flag;
}
int main()
{
	stack<string> stack1;
	stack<string> stack2;
	stack1.push("Geeks");
	stack1.push("4");
	stack1.push("Geeks");
	stack1.push("Welcomes");
	stack1.push("You");
	stack2.push("Geeks");
	stack2.push("4");
	stack2.push("Geeks");
	stack2.push("Welcomes");
```

```
        stack2.push("You");
        if (isSameStack(stack1, stack2))
                cout << "Stacks are Same";
        else
                cout << "Stacks are not Same";
        return 0;
}
```

आउटपुट/Output:

```
Stacks are Same
```

प्रयोग-८ / Experiment-8

क्यू (Queues)

1. ऐरे का उपयोग करके बढ़ने योग्य का प्रोग्राम लिखिए ।
   Write a program to implement growable queues using arrays.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>
#define INITIAL_CAPACITY 5
typedef struct {
   int* arr;
   int front;
   int rear;
   int size;
   int capacity;
} GrowableQueue;
// Function to initialize the queue
void initializeQueue(GrowableQueue* queue) {
   queue->arr = (int*)malloc(INITIAL_CAPACITY * sizeof(int));
   queue->front = -1;
   queue->rear = -1;
   queue->size = 0;
   queue->capacity = INITIAL_CAPACITY;
}
// Function to check if the queue is empty
int isEmpty(GrowableQueue* queue) {
   return queue->size == 0;
}
// Function to enqueue an element
void enqueue(GrowableQueue* queue, int data) {
   if (queue->size == queue->capacity) {
      // Queue is full, double the capacity
      int newCapacity = queue->capacity * 2;
      int* newArr = (int*)malloc(newCapacity * sizeof(int));
      int j = 0;
      for (int i = queue->front; i != queue->rear; i = (i + 1) % queue->capacity) {
```

```
            newArr[j++] = queue->arr[i];
        }
        newArr[j] = queue->arr[queue->rear];
        free(queue->arr);
        queue->arr = newArr;
        queue->front = 0;
        queue->rear = j;
        queue->capacity = newCapacity;
    }
    if (isEmpty(queue)) {
        queue->front = 0;
        queue->rear = 0;
    } else {
        queue->rear = (queue->rear + 1) % queue->capacity;
    }
    queue->arr[queue->rear] = data;
    queue->size++;
}
// Function to dequeue an element
int dequeue(GrowableQueue* queue) {
    if (isEmpty(queue)) {
        printf("Error: Queue is empty\n");
        return -1;
    }
    int data = queue->arr[queue->front];
    if (queue->front == queue->rear) {
        queue->front = -1;
        queue->rear = -1;
    } else {
        queue->front = (queue->front + 1) % queue->capacity;
    }
    queue->size--;
    return data;
}
// Function to display the queue
void display(GrowableQueue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue: ");
    int i = queue->front;
    int count = queue->size;
    while (count > 0) {
        printf("%d ", queue->arr[i]);
        i = (i + 1) % queue->capacity;
        count--;
    }
    printf("\n");
}
```

```
// Function to free the allocated memory
void destroyQueue(GrowableQueue* queue) {
    free(queue->arr);
    queue->arr = NULL;
    queue->front = -1;
    queue->rear = -1;
    queue->size = 0;
    queue->capacity = 0;
}
int main() {
    GrowableQueue queue;
    initializeQueue(&queue);
    enqueue(&queue, 10);
    enqueue(&queue, 20);
    enqueue(&queue, 30);
    enqueue(&queue, 40);
    enqueue(&queue, 50);
    display(&queue);
    enqueue(&queue, 60);
    enqueue(&queue, 70);
    display(&queue);
    printf("Dequeue: %d\n", dequeue(&queue));
    printf("Dequeue: %d\n", dequeue(&queue));
    display(&queue);
    destroyQueue(&queue);
    return 0;
}
```

आउटपुट/Output:

```
Queue: 10 20 30 40 50
Queue: 10 20 30 40 50 60 70
Dequeue: 10
Dequeue: 20
Queue: 30 40 50 60 70
```

2. लिंक्ड लिस्ट का उपयोग करके क्यू बनाए ।
   Implement queues using a linked list.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>
// Node structure
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Queue structure
typedef struct {
    Node* front;
    Node* rear;
} Queue;

// Function to initialize the queue
void initializeQueue(Queue* queue) {
    queue->front = NULL;
    queue->rear = NULL;
}

// Function to check if the queue is empty
int isEmpty(Queue* queue) {
    return queue->front == NULL;
}

// Function to enqueue an element
void enqueue(Queue* queue, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;

    if (isEmpty(queue)) {
        queue->front = newNode;
        queue->rear = newNode;
```

```
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
}

// Function to dequeue an element
int dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Error: Queue is empty\n");
        return -1;
    }

    int data = queue->front->data;
    Node* temp = queue->front;

    if (queue->front == queue->rear) {
        queue->front = NULL;
        queue->rear = NULL;
    } else {
        queue->front = queue->front->next;
    }

    free(temp);
    return data;
}

// Function to display the queue
void display(Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty\n");
        return;
    }

    printf("Queue: ");
    Node* temp = queue->front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Function to free the allocated memory
void destroyQueue(Queue* queue) {
    Node* temp = queue->front;
    while (temp != NULL) {
        Node* nextNode = temp->next;
        free(temp);
        temp = nextNode;
```

```
    }
    queue->front = NULL;
    queue->rear = NULL;
}

int main() {
    Queue queue;
    initializeQueue(&queue);

    enqueue(&queue, 10);
    enqueue(&queue, 20);
    enqueue(&queue, 30);
    enqueue(&queue, 40);
    enqueue(&queue, 50);

    display(&queue);

    enqueue(&queue, 60);
    enqueue(&queue, 70);

    display(&queue);

    printf("Dequeue: %d\n", dequeue(&queue));
    printf("Dequeue: %d\n", dequeue(&queue));
    display(&queue);
    destroyQueue(&queue);
    return 0;
}
```

आउटपुट/Output:

```
Queue: 10 20 30 40 50
Queue: 10 20 30 40 50 60 70
Dequeue: 10
Dequeue: 20
Queue: 30 40 50 60 70
```

3. दो स्टैक का उपयोग करके क्यू का प्रोग्राम लिखिए ।
   Write a program to implement queues using two stacks.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int *data;
    int top;
    int capacity;
} Stack;

// Function to create a new stack
Stack *createStack(int capacity) {
    Stack *stack = (Stack *)malloc(sizeof(Stack));
    stack->data = (int *)malloc(capacity * sizeof(int));
    stack->top = -1;
    stack->capacity = capacity;
    return stack;
}

// Function to check if a stack is empty
int isEmpty(Stack *stack) {
    return stack->top == -1;
}

// Function to check if a stack is full
int isFull(Stack *stack) {
    return stack->top == stack->capacity - 1;
}

// Function to push an element onto the stack
void push(Stack *stack, int item) {
    if (isFull(stack)) {
        printf("Stack overflow\n");
        return;
    }
    stack->data[++stack->top] = item;
}

// Function to pop an element from the stack
int pop(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow\n");
        return -1;
    }
```

```
    return stack->data[stack->top--];
}

// Queue structure using two stacks
typedef struct {
    Stack *enqueueStack;
    Stack *dequeueStack;
} Queue;

// Function to create a new queue
Queue *createQueue(int capacity) {
    Queue *queue = (Queue *)malloc(sizeof(Queue));
    queue->enqueueStack = createStack(capacity);
    queue->dequeueStack = createStack(capacity);
    return queue;
}

// Function to check if the queue is empty
int isEmptyQueue(Queue *queue) {
    return isEmpty(queue->enqueueStack) && isEmpty(queue->dequeueStack);
}

// Function to enqueue an element into the queue
void enqueue(Queue *queue, int item) {
    while (!isEmpty(queue->dequeueStack)) {
        push(queue->enqueueStack, pop(queue->dequeueStack));
    }
    push(queue->enqueueStack, item);
}

// Function to dequeue an element from the queue
int dequeue(Queue *queue) {
    while (!isEmpty(queue->enqueueStack)) {
        push(queue->dequeueStack, pop(queue->enqueueStack));
    }

    if (isEmpty(queue->dequeueStack)) {
        printf("Queue is empty\n");
        return -1;
    }

    return pop(queue->dequeueStack);
}

// Function to free the allocated memory
void destroyQueue(Queue *queue) {
    free(queue->enqueueStack->data);
    free(queue->dequeueStack->data);
    free(queue->enqueueStack);
    free(queue->dequeueStack);
```

```
    free(queue);
}

int main() {
    Queue *queue = createQueue(5);
    enqueue(queue, 10);
    enqueue(queue, 20);
    enqueue(queue, 30);
    printf("Dequeue: %d\n", dequeue(queue));
    enqueue(queue, 40);
    enqueue(queue, 50);
    printf("Dequeue: %d\n", dequeue(queue));
    printf("Dequeue: %d\n", dequeue(queue));
    printf("Dequeue: %d\n", dequeue(queue));
    destroyQueue(queue);
    return 0;
}
```

आउटपुट/Output:

```
Dequeue: 10
Dequeue: 20
Dequeue: 30
Dequeue: 40
```

4. इन्सर्ट और डिलीट फ़ंक्शंस के साथ डबल-एंडेड क्यू बनाए ।
   Implement double-ended queues with insert and delete functions.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

// Node structure for doubly linked list
typedef struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
} Node;

// Deque structure
typedef struct {
   Node* front;
   Node* rear;
} Deque;

// Function to initialize the deque
void initializeDeque(Deque* deque) {
   deque->front = NULL;
   deque->rear = NULL;
}

// Function to check if the deque is empty
int isEmpty(Deque* deque) {
   return deque->front == NULL;
}

// Function to create a new node
Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

// Function to insert an element at the front of the deque
void insertFront(Deque* deque, int data) {
   Node* newNode = createNode(data);

   if (isEmpty(deque)) {
      deque->front = newNode;
      deque->rear = newNode;
   } else {
```

```
        newNode->next = deque->front;
        deque->front->prev = newNode;
        deque->front = newNode;
    }
}

// Function to insert an element at the rear of the deque
void insertRear(Deque* deque, int data) {
    Node* newNode = createNode(data);

    if (isEmpty(deque)) {
        deque->front = newNode;
        deque->rear = newNode;
    } else {
        newNode->prev = deque->rear;
        deque->rear->next = newNode;
        deque->rear = newNode;
    }
}

// Function to delete an element from the front of the deque
int deleteFront(Deque* deque) {
    if (isEmpty(deque)) {
        printf("Error: Deque is empty\n");
        return -1;
    }

    int data = deque->front->data;
    Node* temp = deque->front;

    if (deque->front == deque->rear) {
        deque->front = NULL;
        deque->rear = NULL;
    } else {
        deque->front = deque->front->next;
        deque->front->prev = NULL;
    }

    free(temp);
    return data;
}

// Function to delete an element from the rear of the deque
int deleteRear(Deque* deque) {
    if (isEmpty(deque)) {
        printf("Error: Deque is empty\n");
        return -1;
    }

    int data = deque->rear->data;
```

```
    Node* temp = deque->rear;

    if (deque->front == deque->rear) {
        deque->front = NULL;
        deque->rear = NULL;
    } else {
        deque->rear = deque->rear->prev;
        deque->rear->next = NULL;
    }

    free(temp);
    return data;
}

// Function to display the deque from front to rear
void display(Deque* deque) {
    if (isEmpty(deque)) {
        printf("Deque is empty\n");
        return;
    }

    printf("Deque (Front to Rear): ");
    Node* temp = deque->front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Function to free the allocated memory
void destroyDeque(Deque* deque) {
    Node* temp = deque->front;
    while (temp != NULL) {
        Node* nextNode = temp->next;
        free(temp);
        temp = nextNode;
    }
    deque->front = NULL;
    deque->rear = NULL;
}

int main() {
    Deque deque;
    initializeDeque(&deque);

    insertFront(&deque, 10);
    insertFront(&deque, 20);
    insertRear(&deque, 30);
    insertRear(&deque, 40);
```

```
    display(&deque);

    printf("Delete Front: %d\n", deleteFront(&deque));
    printf("Delete Rear: %d\n", deleteRear(&deque));

    display(&deque);
    destroyDeque(&deque);
    return 0;
}
```

आउटपुट/Output:

```
Deque (Front to Rear): 20 10 30 40
Delete Front: 20
Delete Rear: 40
Deque (Front to Rear): 10 30
```

प्रयोग-९ / Experiment-9

ट्रीज (Trees):

1. बाइनरी ट्री को 1डी ऐरे में संग्रहित करने के लिए एक प्रोग्राम लिखें ।
   Write a program to store binary trees in a 1D array.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

#define ARRAY_SIZE 15

// Structure for binary tree node
typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// Function to build a binary tree from array representation
Node* buildTreeFromArray(int arr[], int index, int size) {
    if (index >= size || arr[index] == -1) {
        return NULL; // Empty node or out of array bounds
    }

    Node* newNode = createNode(arr[index]);
    newNode->left = buildTreeFromArray(arr, 2 * index + 1, size);
    newNode->right = buildTreeFromArray(arr, 2 * index + 2, size);
    return newNode;
}

// Function to print the binary tree using in-order traversal
void inOrderTraversal(Node* root) {
    if (root == NULL) {
        return;
    }
    inOrderTraversal(root->left);
    printf("%d ", root->data);
```

```
    inOrderTraversal(root->right);
}

int main() {
    int arr[ARRAY_SIZE] = {1, 2, 3, 4, 5, -1, 6, -1, -1, 7, 8}; // Binary tree represented in
array

    Node* root = buildTreeFromArray(arr, 0, ARRAY_SIZE);

    printf("Original Binary Tree (In-order traversal): ");
    inOrderTraversal(root);
    printf("\n");

    // Converting binary tree to array representation
    int newArray[ARRAY_SIZE];
    for (int i = 0; i < ARRAY_SIZE; i++) {
        newArray[i] = -1; // Initialize with empty nodes
    }

    // Helper function to convert binary tree to array
    void treeToArray(Node* node, int index) {
        if (node == NULL) {
            return;
        }
        newArray[index] = node->data;
        treeToArray(node->left, 2 * index + 1);
        treeToArray(node->right, 2 * index + 2);
    }

    treeToArray(root, 0);

    // Printing the array representation of the binary tree
    printf("Array Representation of Binary Tree: ");
    for (int i = 0; i < ARRAY_SIZE; i++) {
        if (newArray[i] != -1) {
            printf("%d ", newArray[i]);
        }
    }
    printf("\n");

    // Don't forget to free the allocated memory after use
    free(root);

    return 0;
}
```

आउटपुट/Output:

```
Original Binary Tree (In-order traversal): 4 2 7 5 8 1 3 0 6 0
Array Representation of Binary Tree: 1 2 3 4 5 6 7 8 0 0
```

2. बाइनरी ट्री को लिंक्ड लिस्ट में संग्रहित करने के लिए एक प्रोग्राम लिखें ।
   Write a program to store binary trees in the linked list.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

// Structure to represent a binary tree node
struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Structure to represent a linked list node
struct ListNode {
    struct TreeNode* treeNode;
    struct ListNode* next;
};

// Function to create a new binary tree node
struct TreeNode* createTreeNode(int data) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// Function to create a new linked list node
struct ListNode* createListNode(struct TreeNode* treeNode) {
    struct ListNode* newNode = (struct ListNode*)malloc(sizeof(struct ListNode));
    newNode->treeNode = treeNode;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a tree node into the linked list
void insert(struct ListNode** head, struct TreeNode* treeNode) {
    struct ListNode* newNode = createListNode(treeNode);
    newNode->next = *head;
    *head = newNode;
```

```
}

// Function to display the linked list
void display(struct ListNode* head) {
    struct ListNode* current = head;
    while (current != NULL) {
        printf("%d ", current->treeNode->data);
        current = current->next;
    }
    printf("\n");
}

int main() {
    struct TreeNode* root = createTreeNode(1);
    root->left = createTreeNode(2);
    root->right = createTreeNode(3);
    root->left->left = createTreeNode(4);
    root->left->right = createTreeNode(5);
    root->right->left = createTreeNode(6);
    root->right->right = createTreeNode(7);

    struct ListNode* listHead = NULL;
    insert(&listHead, root->right->right);
    insert(&listHead, root->right->left);
    insert(&listHead, root->right);
    insert(&listHead, root->left->right);
    insert(&listHead, root->left->left);
    insert(&listHead, root->left);
    insert(&listHead, root);

    printf("Linked list elements (in reverse order of tree traversal): ");
    display(listHead);

    return 0;
}
```

आउटपुट/Output:

Linked list elements (in reverse order of tree traversal): 7 6 3 5 4 2 1

3. बाइनरी सर्च ट्री के 1डी ऐरे कार्यान्वयन में किसी तत्व को उसकी कुंजी दिए जाने का कार्यक्रम लिखें ।
   Write a program for finding an element given its key in a 1D array implementation of binary search trees.

समाधान/Solution:

```
#include <stdio.h>
#include <stdbool.h>

// Function to search for an element in a binary search tree
bool searchElement(int bst[], int size, int key) {
    int current = 0;

    while (current < size) {
        if (bst[current] == key) {
            return true; // Element found
        } else if (key < bst[current]) {
            current = 2 * current + 1; // Move to the left child
        } else {
            current = 2 * current + 2; // Move to the right child
        }
    }
    return false; // Element not found
}

int main() {
    int bst[] = {15, 10, 20, 5, 12, 17, 25};
    int size = sizeof(bst) / sizeof(bst[0]);

    int key;
    printf("Enter the key to search: ");
    scanf("%d", &key);

    if (searchElement(bst, size, key)) {
        printf("Element %d found in the binary search tree.\n", key);
    } else {
        printf("Element %d not found in the binary search tree.\n", key);
    }

    return 0;
}
```

आउटपुट/Output:

Enter the key to search: 17
Element 17 found in the binary search tree.

4. रिकर्सन का उपयोग करके बी.एस.टी. के लिंक्ड लिस्ट कार्यान्वयन में इसकी कुंजी दिए गए तत्व को खोजने का कार्यक्रम लिखें ।

   Write a program for finding the element given its key in a linked list implementation of B.S.T. using recursion.

समाधान/Solution:

```
#include <stdio.h>
#include <stdlib.h>

// Structure for a binary search tree node
struct TreeNode {
   int key;
   struct TreeNode* left;
   struct TreeNode* right;
};

// Function to create a new node
struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
   newNode->key = key;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}

// Function to insert a key into the BST
struct TreeNode* insert(struct TreeNode* root, int key) {
   if (root == NULL) {
      return createNode(key);
   }

   if (key < root->key) {
      root->left = insert(root->left, key);
   } else if (key > root->key) {
      root->right = insert(root->right, key);
   }

   return root;
}

// Function to search for a key in the BST using recursion
struct TreeNode* search(struct TreeNode* root, int key) {
   if (root == NULL || root->key == key) {
      return root;
   }

   if (key < root->key) {
      return search(root->left, key);
   }
```

```
    return search(root->right, key);
}

int main() {
    struct TreeNode* root = NULL;

    root = insert(root, 15);
    insert(root, 10);
    insert(root, 20);
    insert(root, 5);
    insert(root, 12);
    insert(root, 17);
    insert(root, 25);

    int key;
    printf("Enter the key to search: ");
    scanf("%d", &key);

    struct TreeNode* result = search(root, key);

    if (result != NULL) {
        printf("Element %d found in the binary search tree.\n", key);
    } else {
        printf("Element %d not found in the binary search tree.\n", key);
    }

    return 0;
}
```

आउटपुट/Output:

```
Enter the key to search: 17
Element 17 found in the binary search tree.
```

प्रयोग-१० / Experiment-10

ट्रीज (Trees):

5. बी.एस.टी. की अधिकतम डेप्थ ज्ञात करने के लिए एक प्रोग्राम लिखिए ।
   Write a program to find the maximum depth of the BST.

समाधान/Solution

```
#include <stdio.h>
#include <stdlib.h>

// Structure for a binary search tree node
struct TreeNode {
   int key;
   struct TreeNode* left;
   struct TreeNode* right;
};

// Function to create a new node
struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
   newNode->key = key;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}

// Function to insert a key into the BST
struct TreeNode* insert(struct TreeNode* root, int key) {
   if (root == NULL) {
      return createNode(key);
   }

   if (key < root->key) {
      root->left = insert(root->left, key);
   } else if (key > root->key) {
      root->right = insert(root->right, key);
   }

   return root;
}

// Function to calculate the maximum depth of the BST
int maxDepth(struct TreeNode* root) {
   if (root == NULL) {
      return 0;
   } else {
      int leftDepth = maxDepth(root->left);
```

```
        int rightDepth = maxDepth(root->right);
        return 1 + (leftDepth > rightDepth ? leftDepth : rightDepth);
    }
}

int main() {
    struct TreeNode* root = NULL;

    root = insert(root, 15);
    insert(root, 10);
    insert(root, 20);
    insert(root, 5);
    insert(root, 12);
    insert(root, 17);
    insert(root, 25);

    int depth = maxDepth(root);
    printf("Maximum depth of the BST: %d\n", depth);

    return 0;
}
```

आउटपुट/Output:

Maximum depth of the BST: 3

6.बी.एस.टी. को इनऑर्डर, प्रीऑर्डर, पोस्टऑर्डर में रिकर्शन के साथ और उसके बिना ट्रैवर्स कीजिये ।
Traverse the BST in Inorder, Preorder, Postorder with and without recursion.

समाधान/Solution

```
#include <stdio.h>
#include <stdlib.h>

// Structure for a binary search tree node
struct TreeNode {
    int key;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Function to create a new node
struct TreeNode* createNode(int key) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    newNode->key = key;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// Function to insert a key into the BST
struct TreeNode* insert(struct TreeNode* root, int key) {
    if (root == NULL) {
        return createNode(key);
    }

    if (key < root->key) {
        root->left = insert(root->left, key);
    } else if (key > root->key) {
        root->right = insert(root->right, key);
    }

    return root;
}

// Inorder traversal using recursion
void inorderRecursive(struct TreeNode* root) {
    if (root != NULL) {
        inorderRecursive(root->left);
        printf("%d ", root->key);
```

```
        inorderRecursive(root->right);
    }
}

// Preorder traversal using recursion
void preorderRecursive(struct TreeNode* root) {
    if (root != NULL) {
        printf("%d ", root->key);
        preorderRecursive(root->left);
        preorderRecursive(root->right);
    }
}

// Postorder traversal using recursion
void postorderRecursive(struct TreeNode* root) {
    if (root != NULL) {
        postorderRecursive(root->left);
        postorderRecursive(root->right);
        printf("%d ", root->key);
    }
}

int main() {
    struct TreeNode* root = NULL;

    root = insert(root, 15);
    insert(root, 10);
    insert(root, 20);
    insert(root, 5);
    insert(root, 12);
    insert(root, 17);
    insert(root, 25);

    printf("Inorder traversal (recursive): ");
    inorderRecursive(root);
    printf("\n");

    printf("Preorder traversal (recursive): ");
    preorderRecursive(root);
    printf("\n");

    printf("Postorder traversal (recursive): ");
    postorderRecursive(root);
```

```
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

Inorder traversal (recursive): 5 10 12 15 17 20 25
Preorder traversal (recursive): 15 10 5 12 20 17 25
Postorder traversal (recursive): 5 12 10 17 25 20 15

7.किसी दिए गए अवर्गीकृत ऐरे से उसका संगत बी.एस.टी. बनाएं और सभी तत्वों को क्रमबद्ध प्रिंट करें (अर्थात इनऑर्डर ट्रैवर्सल) ।

From a given unsorted array create its corresponding BST and print all the elements in sorted order (i.e. inorder traversal).

समाधान/Solution

```
#include <stdio.h>
#include <stdlib.h>

// Structure for a binary search tree node
struct TreeNode {
    int key;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Function to create a new node
struct TreeNode* createNode(int key) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    newNode->key = key;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// Function to insert a key into the BST
struct TreeNode* insert(struct TreeNode* root, int key) {
    if (root == NULL) {
        return createNode(key);
    }

    if (key < root->key) {
        root->left = insert(root->left, key);
    } else if (key > root->key) {
        root->right = insert(root->right, key);
    }

    return root;
}

// Inorder traversal to print elements in sorted order
void inorder(struct TreeNode* root) {
```

```
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}

int main() {
    int unsortedArray[] = {15, 10, 20, 5, 12, 17, 25};
    int size = sizeof(unsortedArray) / sizeof(unsortedArray[0]);

    struct TreeNode* root = NULL;

    for (int i = 0; i < size; i++) {
        root = insert(root, unsortedArray[i]);
    }

    printf("Sorted elements using inorder traversal: ");
    inorder(root);
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

Sorted elements using inorder traversal: 5 10 12 15 17 20 25

प्रयोग-११ / Experiment-11

ग्राफ (Graphs)

1. ऐडजसेनसी मैट्रिक्स, इंसीडेंस मैट्रिक्स, ऐडजसेनसी सूची और डाइरेक्ट रिप्रजेंटेशन का उपयोग कर ग्राफ इम्प्लीमेंट करें ।

   Implement graphs using an adjacency matrix, incidence matrix, adjacency list, and direct representation.

समाधान/Solution

```
#include <stdio.h>

#define MAX_NODES 10

int adjacencyMatrix[MAX_NODES][MAX_NODES];

int main() {
    int numNodes, numEdges;
    printf("Enter the number of nodes and edges: ");
    scanf("%d %d", &numNodes, &numEdges);

    for (int i = 0; i < numEdges; i++) {
        int u, v;
        printf("Enter edge %d (source destination): ", i + 1);
        scanf("%d %d", &u, &v);
        adjacencyMatrix[u][v] = 1;
        adjacencyMatrix[v][u] = 1; // For undirected graph
    }

    printf("Adjacency Matrix:\n");
    for (int i = 0; i < numNodes; i++) {
        for (int j = 0; j < numNodes; j++) {
            printf("%d ", adjacencyMatrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

आउटपुट/Output:

Enter the number of nodes and edges: 4 3

Enter edge 1 (source destination): 0 1

Enter edge 2 (source destination): 1 2

Enter edge 3 (source destination): 2 3

Adjacency Matrix:

0 1 0 0

1 0 1 0

0 1 0 1

0 0 1 0

2. ग्राफ के एडजसेन्ट मैट्रिक्स इम्प्लीमेंटेशन पर डेप्थ-फर्स्ट आर्डर में ग्राफ को ट्रैवर्स करें ।
   Traverse the graph in Depth-first order on an adjacent matrix implementation of the graph.

समाधान/Solution

```
#include <stdio.h>

#define MAX_NODES 10

int adjacencyMatrix[MAX_NODES][MAX_NODES];
int visited[MAX_NODES];

void dfs(int node, int numNodes) {
   visited[node] = 1;
   printf("%d ", node);

   for (int i = 0; i < numNodes; i++) {
      if (adjacencyMatrix[node][i] == 1 && !visited[i]) {
         dfs(i, numNodes);
      }
   }
}

int main() {
   int numNodes, numEdges;
   printf("Enter the number of nodes and edges: ");
   scanf("%d %d", &numNodes, &numEdges);

   for (int i = 0; i < numEdges; i++) {
      int u, v;
      printf("Enter edge %d (source destination): ", i + 1);
      scanf("%d %d", &u, &v);
      adjacencyMatrix[u][v] = 1;
      adjacencyMatrix[v][u] = 1; // For undirected graph
   }

   printf("DFS Traversal starting from node 0: ");
   dfs(0, numNodes);
   printf("\n");

   return 0;
}
```

आउटपुट/Output:

Enter the number of nodes and edges: 6 7

Enter edge 1 (source destination): 0 1

Enter edge 2 (source destination): 0 2

Enter edge 3 (source destination): 1 3

Enter edge 4 (source destination): 1 4

Enter edge 5 (source destination): 2 5

Enter edge 6 (source destination): 3 4

Enter edge 7 (source destination): 4 5

DFS Traversal starting from node 0: 0 1 3 4 5 2

3. ग्राफ़ के एडजसेन्ट मैट्रिक्स इम्प्लीमेंटेशन पर ब्रेड्थ-फर्स्ट आर्डर में ग्राफ को ट्रैवर्स करें ।
   Traverse the graph in Breadth-first order on an adjacent matrix implementation of the graph.

समाधान/Solution

```
#include <stdio.h>
#include <stdbool.h>

#define MAX_NODES 10

int adjacencyMatrix[MAX_NODES][MAX_NODES];
int visited[MAX_NODES];
int queue[MAX_NODES];
int front = -1, rear = -1;

void enqueue(int node) {
   if (rear == MAX_NODES - 1) {
      printf("Queue is full.\n");
   } else {
      if (front == -1) {
         front = 0;
      }
      rear++;
      queue[rear] = node;
   }
}

int dequeue() {
   if (front == -1) {
      printf("Queue is empty.\n");
      return -1;
   } else {
      int node = queue[front];
      front++;
      if (front > rear) {
         front = rear = -1;
      }
      return node;
   }
}

bool isQueueEmpty() {
   return front == -1;
}

void bfs(int startNode, int numNodes) {
```

```
    enqueue(startNode);
    visited[startNode] = 1;

    while (!isQueueEmpty()) {
        int currentNode = dequeue();
        printf("%d ", currentNode);

        for (int i = 0; i < numNodes; i++) {
            if (adjacencyMatrix[currentNode][i] == 1 && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
}

int main() {
    int numNodes, numEdges;
    printf("Enter the number of nodes and edges: ");
    scanf("%d %d", &numNodes, &numEdges);

    for (int i = 0; i < numEdges; i++) {
        int u, v;
        printf("Enter edge %d (source destination): ", i + 1);
        scanf("%d %d", &u, &v);
        adjacencyMatrix[u][v] = 1;
        adjacencyMatrix[v][u] = 1; // For undirected graph
    }

    printf("BFS Traversal starting from node 0: ");
    bfs(0, numNodes);
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

```
Enter the number of nodes and edges: 6 7
Enter edge 1 (source destination): 0 1
Enter edge 2 (source destination): 0 2
Enter edge 3 (source destination): 1 3
Enter edge 4 (source destination): 1 4
Enter edge 5 (source destination): 2 5
Enter edge 6 (source destination): 3 4
Enter edge 7 (source destination): 4 5
BFS Traversal starting from node 0: 0 1 2 3 4 5
```

4. डी.एफ.एस. का उपयोग करके अनडायरेक्टेड ग्राफ़ में एक सायकल डिटेक्ट करने के लिए प्रोग्राम लिखें ।

Program for detecting a cycle in an undirected graph using DFS.

```
#include <stdio.h>
#include <stdbool.h>

#define MAX_NODES 10

int adjacencyMatrix[MAX_NODES][MAX_NODES];
int visited[MAX_NODES];

bool hasCycle(int currentNode, int parent, int numNodes) {
    visited[currentNode] = 1;

    for (int i = 0; i < numNodes; i++) {
        if (adjacencyMatrix[currentNode][i] == 1) {
            if (!visited[i]) {
                if (hasCycle(i, currentNode, numNodes)) {
                    return true;
                }
            } else if (i != parent) {
                return true;
            }
        }
    }
    return false;
}

bool containsCycle(int numNodes) {
    for (int i = 0; i < numNodes; i++) {
        if (!visited[i]) {
            if (hasCycle(i, -1, numNodes)) {
                return true;
            }
        }
    }
    return false;
}

int main() {
    int numNodes, numEdges;
    printf("Enter the number of nodes and edges: ");
    scanf("%d %d", &numNodes, &numEdges);

    for (int i = 0; i < numEdges; i++) {
```

```
        int u, v;
        printf("Enter edge %d (source destination): ", i + 1);
        scanf("%d %d", &u, &v);
        adjacencyMatrix[u][v] = 1;
        adjacencyMatrix[v][u] = 1; // For undirected graph
    }

    if (containsCycle(numNodes)) {
        printf("The graph contains a cycle.\n");
    } else {
        printf("The graph does not contain a cycle.\n");
    }

    return 0;
}
```

आउटपुट/Output:

```
Enter the number of nodes and edges: 6 6
Enter edge 1 (source destination): 0 1
Enter edge 2 (source destination): 0 2
Enter edge 3 (source destination): 1 3
Enter edge 4 (source destination): 2 4
Enter edge 5 (source destination): 3 5
Enter edge 6 (source destination): 4 5
The graph contains a cycle.
```

प्रयोग-१२ / Experiment-12

सॉर्टिंग (Sorting)

1. मर्ज सॉर्ट को रिकर्सिवली इम्प्लीमेंट करें।
   Implement Merge sort recursively.

समाधान/Solution

```
#include <stdio.h>

// Function to merge two sorted arrays
void merge(int arr[], int left, int mid, int right) {
   int n1 = mid - left + 1;
   int n2 = right - mid;

   int L[n1], R[n2];

   for (int i = 0; i < n1; i++) {
      L[i] = arr[left + i];
   }
   for (int j = 0; j < n2; j++) {
      R[j] = arr[mid + 1 + j];
   }

   int i = 0, j = 0, k = left;

   while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {
         arr[k] = L[i];
         i++;
      } else {
         arr[k] = R[j];
         j++;
      }
      k++;
   }

   while (i < n1) {
      arr[k] = L[i];
      i++;
      k++;
   }
```

```
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Merge Sort function
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        merge(arr, left, mid, right);
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int arrSize = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    for (int i = 0; i < arrSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    mergeSort(arr, 0, arrSize - 1);

    printf("Sorted array: ");
    for (int i = 0; i < arrSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

```
Original array: 12 11 13 5 6 7

Sorted array: 5 6 7 11 12 13
```

2. क्विक सॉर्ट को रिकर्सिवली इम्प्लीमेंट करें।
   Implement Quick sort recursively.

समाधान/Solution

```
#include <stdio.h>

// Function to partition the array
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

// Quick Sort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int arrSize = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    for (int i = 0; i < arrSize; i++) {
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");

    quickSort(arr, 0, arrSize - 1);

    printf("Sorted array: ");
    for (int i = 0; i < arrSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

Original array: 10 7 8 9 1 5

Sorted array: 1 5 7 8 9 10

3. इंसर्शन सॉर्ट इम्प्लीमेंट करें।
   Implement Insertion sort.

समाधान/Solution

```
#include <stdio.h>

// Insertion Sort function
void insertionSort(int arr[], int arrSize) {
    for (int i = 1; i < arrSize; i++) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6};
    int arrSize = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    for (int i = 0; i < arrSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    insertionSort(arr, arrSize);

    printf("Sorted array: ");
    for (int i = 0; i < arrSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

Original array: 12 11 13 5 6

Sorted array: 5 6 11 12 13

4. सेलेक्शन सॉर्ट इम्प्लीमेंट करें।
   Implement Selection sort.

समाधान/Solution

```
#include <stdio.h>

// Selection Sort function
void selectionSort(int arr[], int arrSize) {
   for (int i = 0; i < arrSize - 1; i++) {
      int minIndex = i;

      for (int j = i + 1; j < arrSize; j++) {
         if (arr[j] < arr[minIndex]) {
            minIndex = j;
         }
      }

      int temp = arr[minIndex];
      arr[minIndex] = arr[i];
      arr[i] = temp;
   }
}

int main() {
   int arr[] = {64, 25, 12, 22, 11};
   int arrSize = sizeof(arr) / sizeof(arr[0]);

   printf("Original array: ");
   for (int i = 0; i < arrSize; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
  selectionSort(arr, arrSize);
  printf("Sorted array: ");
      for (int i = 0; i < arrSize; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}
```

आउटपुट/Output:

```
Original array: 64 25 12 22 11
```

Sorted array: 11 12 22 25 64

5. रेडिक्स सॉर्ट इम्प्लीमेंट करें।
   Implement Radix sort.

समाधान/Solution

```
#include <stdio.h>

// Function to get the maximum value in an array
int getMax(int arr[], int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

// Function to perform counting sort based on the digit at exp
void countingSort(int arr[], int n, int exp) {
    int output[n];
    int count[10] = {0};

    for (int i = 0; i < n; i++) {
        count[(arr[i] / exp) % 10]++;
    }

    for (int i = 1; i < 10; i++) {
        count[i] += count[i - 1];
    }

    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }

    for (int i = 0; i < n; i++) {
        arr[i] = output[i];
    }
}

// Radix Sort function
void radixSort(int arr[], int n) {
    int max = getMax(arr, n);

    for (int exp = 1; max / exp > 0; exp *= 10) {
```

```
        countingSort(arr, n, exp);
    }
}

int main() {
    int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};
    int arrSize = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    for (int i = 0; i < arrSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    radixSort(arr, arrSize);

    printf("Sorted array: ");
    for (int i = 0; i < arrSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

Original array: 170 45 75 90 802 24 2 66

Sorted array: 2 24 45 66 75 90 170 802

प्रयोग-१३ / Experiment-13

(Miscellaneous Problems)

1. N लोगों ने खुद को एक सर्कल में व्यवस्थित करके सर्किल के अराउंड प्रत्येक M-वें व्यक्ति को बाहर निकालकर एक नेता का चुनाव करने का फैसला किया है, प्रत्येक व्यक्ति के बाहर निकलने पर रैंक क्लोज़ हो जाती है। ज्ञात कीजिए कि कौन सा व्यक्ति अंतिम बचेगा (रैंक 1 के साथ) [सर्कुलर लिंक्ड लिस्ट का उपयोग करें] ।

   N people have decided to elect a leader by arranging themselves in a circle and eliminating every M^{th} person around the circle, closing ranks as each person drops out. Find which person will be the last one remaining (with rank 1) [Use Circular Linked List].

समाधान/Solution

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = newNode;  // Point to itself initially
    return newNode;
}

void appendNode(struct Node** head, int data) {
    struct Node* newNode = createNode(data);

    if (*head == NULL) {
        *head = newNode;
        return;
    }

    struct Node* last = *head;
    while (last->next != *head) {
        last = last->next;
    }

    last->next = newNode;
    newNode->next = *head;
```

```
}

void eliminatePerson(struct Node** head, int m) {
    if (*head == NULL) {
        return;
    }

    struct Node* current = *head;
    struct Node* prev;

    while (current->next != current) {
        for (int i = 1; i < m; i++) {
            prev = current;
            current = current->next;
        }

        prev->next = current->next;
        free(current);
        current = prev->next;
    }

    *head = current;
}

int findLastPerson(struct Node* head) {
    return head->data;
}

int main() {
    int N, M;
    printf("Enter the number of people (N) and the value of M: ");
    scanf("%d %d", &N, &M);

    struct Node* circle = NULL;

    for (int i = 1; i <= N; i++) {
        appendNode(&circle, i);
    }

    eliminatePerson(&circle, M);

    int lastPerson = findLastPerson(circle);

    printf("The last person remaining (with rank 1) is: %d\n", lastPerson);

    return 0;
}
```

आउटपुट/Output:

Enter the number of people (N) and the value of M: 7 3
The last person remaining (with rank 1) is: 4

2. कृपया निम्नलिखित समीकरण का उपयोग करके C, U, B, E, D का मान ज्ञात करें:-
(C+U+B+E+D)^3= CUBED, जहां C, U, B, E, D इन्टीजर्स हैं।

Please find the value of C, U, B, E, and D using the following equation:-
(C+U+B+E+D)^3= CUBED, where C, U, B, E, and D are integers.

समाधान/Solution

```
#include <stdio.h>
#include <math.h>

int main() {
   for (int C = 0; C <= 9; C++) {
      for (int U = 0; U <= 9; U++) {
         for (int B = 0; B <= 9; B++) {
            for (int E = 0; E <= 9; E++) {
               for (int D = 0; D <= 9; D++) {
                  int left_side = pow((C + U + B + E + D), 3);
                  int right_side = (C * 10000) + (U * 1000) + (B * 100) + (E * 10) + D;

                  if (left_side == right_side) {
                     printf("C = %d, U = %d, B = %d, E = %d, D = %d\n", C, U, B, E, D);
                  }
               }
            }
         }
      }
   }

   return 0;
}
```

आउटपुट/Output:

C = 2, U = 1, B = 8, E = 3, D = 9

प्रयोग-१४ / Experiment-14

(Miscellaneous Problems)

1. स्टैक को इस तरह से इम्प्लीमेंट करें कि पुश, पॉप और गेट मिनिमम ऑपरेशंस में कॉन्सटैंट समय लगे। [दो स्टैक का उपयोग करें]
   Implement Stack in a way such that Push, Pop, and Get Minimum operations take constant time. [Use two stacks]

समाधान/Solution

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 100

struct Stack {
   int arr[MAX_SIZE];
   int top;
   int minStack[MAX_SIZE];
   int minTop;
};

void initialize(struct Stack* stack) {
   stack->top = -1;
   stack->minTop = -1;
}

int isEmpty(struct Stack* stack) {
   return stack->top == -1;
}

void push(struct Stack* stack, int value) {
   if (stack->top == MAX_SIZE - 1) {
      printf("Stack overflow\n");
      return;
   }

   stack->top++;
   stack->arr[stack->top] = value;

   if (stack->minTop == -1 || value <= stack->minStack[stack->minTop]) {
      stack->minTop++;
      stack->minStack[stack->minTop] = value;
   }
}
```

```
int pop(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow\n");
        return -1;
    }

    int popped = stack->arr[stack->top];
    stack->top--;

    if (popped == stack->minStack[stack->minTop]) {
        stack->minTop--;
    }

    return popped;
}

int getMinimum(struct Stack* stack) {
    if (stack->minTop == -1) {
        printf("Stack is empty\n");
        return -1;
    }

    return stack->minStack[stack->minTop];
}

int main() {
    struct Stack stack;
    initialize(&stack);

    push(&stack, 3);
    push(&stack, 5);
    push(&stack, 2);
    push(&stack, 7);

    printf("Minimum element: %d\n", getMinimum(&stack));
    printf("Popped element: %d\n", pop(&stack));
    printf("Minimum element: %d\n", getMinimum(&stack));

    return 0;
}
```

आउटपुट/Output:

Minimum element: 2

Popped element: 7

Minimum element: 2

2. हमें N दिनों के लिए स्टॉक की कीमतों की एक सूची दी गई है। हमें प्रत्येक दिन के लिए अवधि ज्ञात करने की आवश्यकता है। स्पैन को उस दिन से पहले लगातार दिनों की संख्या के रूप में परिभाषित किया गया है जहां स्टॉक की कीमत दिए गए दिन की कीमत से कम या उसके बराबर थी। उदाहरण के लिए, {100, 60,70,65, 80, 85} स्पैन {1, 1, 2, 1, 4, 5} होगा। पहले दिन के लिए स्पैन हमेशा 1 होता है। उदाहरण में हम देख सकते हैं कि दूसरे दिन के लिए 60 , इससे पहले कोई दिन ऐसा नहीं था जब कीमत 60 से कम थी। इसलिए स्पैन फिर से 1 है। दिन 3 के लिए, दिन 2 (60) पर कीमत 70 से कम है, इसलिए स्पैन 2 है। इसी तरह, दिन 4 और दिन 5 के लिए है । याद रखें कि दिन लगातार होने चाहिए, कि दिन 4 के लिए स्पैन 1 क्यों है, भले ही एक दिन 2 था जहां कीमत 65 से कम थी। आशा है कि यह उदाहरण समस्या को स्पष्ट करता है।

We are given a list of prices of a stock for N number of days. We need to find the span for each day. Span is defined as a number of consecutive days before the given day where the stock price was less than or equal to the price on a given day. For example, {100, 60,70,65, 80, 85} span will be {1, 1, 2, 1, 4, 5}. For the first day, the span is always 1. For example, we can see that for day 2 at 60, there is no day before it where the price was less than 60. Hence span is 1 again. For day 3, the price at day 2 (60) is less than 70, hence span is 2. Similarly, for day 4 and day 5. Remember days should be consecutive, that's why span for day 4 is 1 even though there was a day 2 where price was less than 65. Hope this example clarifies the problem.

समाधान/Solution

```
#include <stdio.h>
#include <stdlib.h>

void calculateSpan(int prices[], int n, int spans[]) {
    spans[0] = 1;

    for (int i = 1; i < n; i++) {
        spans[i] = 1;

        for (int j = i - 1; j >= 0 && prices[i] >= prices[j]; j--) {
            spans[i]++;
        }
    }
}

int main() {
    int n;
    printf("Enter the number of days: ");
    scanf("%d", &n);

    int prices[n];
    for (int i = 0; i < n; i++) {
        printf("Enter the price for day %d: ", i + 1);
```

```
        scanf("%d", &prices[i]);
    }

    int spans[n];
    calculateSpan(prices, n, spans);

    printf("Spans for each day: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", spans[i]);
    }
    printf("\n");

    return 0;
}
```

आउटपुट/Output:

```
Enter the number of days: 6
Enter the price for day 1: 100
Enter the price for day 2: 60
Enter the price for day 3: 70
Enter the price for day 4: 65
Enter the price for day 5: 80
Enter the price for day 6: 85
Spans for each day: 1 1 2 1 4 5
```

References:

1. Data Structures and Algorithms, Annotated Reference with Examples First Edition
2. Data Structures book by G.S. Baluja
3. Data Structures and Algorithms Made Easy.
4. Data structures with c - by schaum series 2