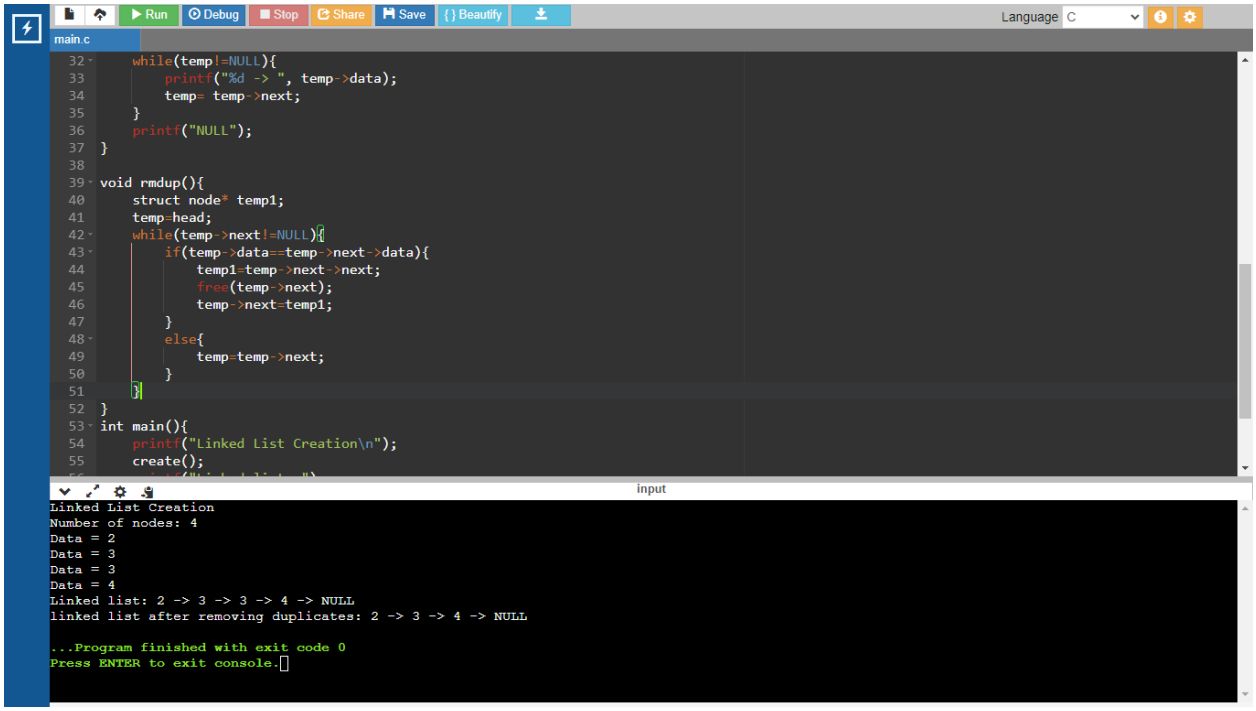
ADVANCED C PROGRAMMING – MODULE 2

1. **Write a C program to remove duplicate element from sorted Linked List.**

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

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

struct node *head, *temp, *newnode;

void create(){
   int n;
   printf("Number of nodes: ");
   scanf("%d", &n);
   for(int i=0;i<n;i++){
      newnode = (struct node *)malloc(sizeof(struct node));
      printf("Data = ");
      scanf("%d", &newnode->data);
      newnode->next = NULL;
      if(head==NULL){
         head=temp=newnode;
      }
      else{
         temp->next=newnode;
         temp=newnode;
      }
   }
}

void display(){
   temp=head;
   while(temp!=NULL){
      printf("%d -> ", temp->data);
      temp= temp->next;
   }
   printf("NULL");
}

void rmdup(){
   struct node* temp1;
   temp=head;
   while(temp->next!=NULL){
      if(temp->data==temp->next->data){
         temp1=temp->next->next;
         free(temp->next);
         temp->next=temp1;
      }
```

```
        else{
            temp=temp->next;
        }
    }
}
int main(){
    printf("Linked List Creation\n");
    create();
    printf("Linked list: ");
    display();
    rmdup();
    printf("\nlinked list after removing duplicates: ");
    display();

}
```

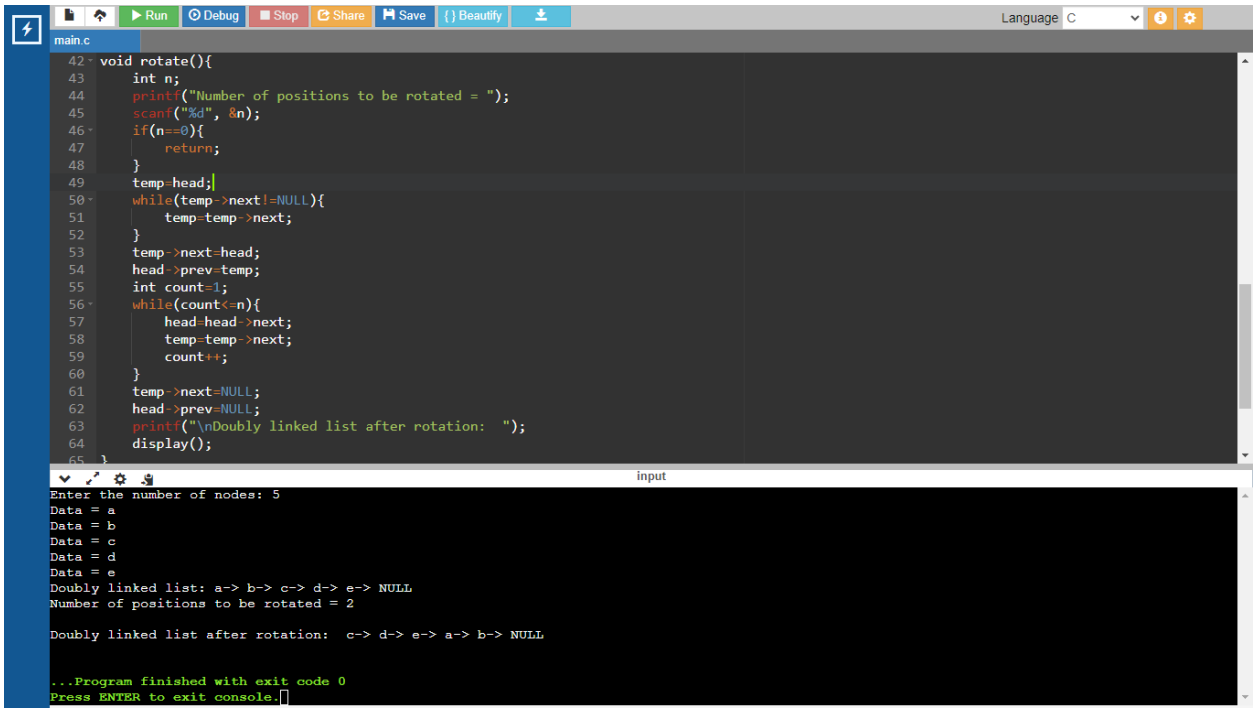


2. Write a C program to rotate a doubly linked list by N nodes.

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

struct node{
    char data;
    struct node* prev;
    struct node* next;
};

struct node *head, *temp, *newnode;

void create(){
    int n;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
```

```
    for(int i=0;i<n;i++){
        newnode = (struct node *)malloc(sizeof(struct node));
        printf("Data = ");
        scanf(" %c", &newnode->data);
        newnode->prev = NULL;
        newnode->next = NULL;
        if(head==NULL){
            head=temp=newnode;
        }
        else{
            temp->next=newnode;
            newnode->prev=temp;
            temp=newnode;
        }
    }
}

void display(){
    temp=head;
    while(temp!=NULL){
        printf("%c-> ", temp->data);
        temp= temp->next;
    }
    printf("NULL\n");
}

void rotate(){
    int n;
    printf("Number of positions to be rotated = ");
    scanf("%d", &n);
    if(n==0){
        return;
    }
    temp=head;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=head;
    head->prev=temp;
    int count=1;
    while(count<=n){
        head=head->next;
        temp=temp->next;
        count++;
    }
    temp->next=NULL;
    head->prev=NULL;
    printf("\nDoubly linked list after rotation:  ");
    display();
```

```
}
int main(){
   create();
   printf("Doubly linked list: ");
   display();
   rotate();
}
```



3. Write a C program to sort the elements of a queue in ascending order.

```
#include <stdio.h>
#define N 5
int queue[N];
int front = -1;
int rear = -1;

void enqueue(int x){
   if(rear==N-1){
      printf("overflow\n");
   }
   else if((front==-1)&&(rear==-1)){
      front=rear=0;
      queue[rear]=x;
   }
   else{
      rear++;
      queue[rear]=x;
   }
}

void dequeue(){
   if((front==-1)&&(rear==-1)){
```

```
        printf("\nUnderflow");
    }
    else if(front==rear){
        front=rear=-1;
    }
    else{
        front++;
    }
}

void display(){
    if((front==-1)&&(rear==-1)){
        printf("\nQueue is empty");
    }
    else{
        for(int i=front;i<rear+1;i++){
            printf("%d  ", queue[i]);
        }
    }
}

void sort(){
    int n=rear - front +1;
    int i, j, temp;
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(queue[i]>queue[j]){
                temp=queue[i];
                queue[i]=queue[j];
                queue[j]=temp;
            }
        }
    }
}
void main(){
    enqueue(4);
    enqueue(2);
    enqueue(7);
    enqueue(5);
    enqueue(1);
    printf("\nBefore sorting:  ");
    display();
    sort();
    printf("\nAfter sorting:  ");
    display();

}
```

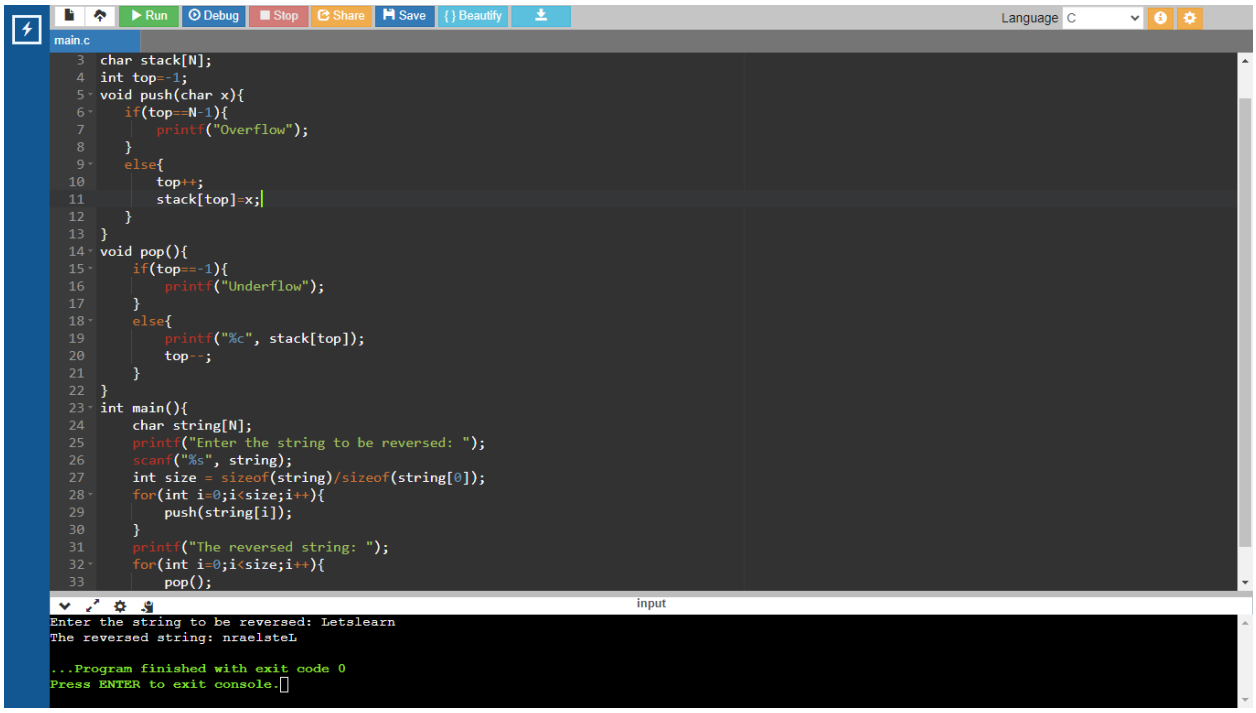
4. **List all queue function operations available for manipulation of data elements in C**

enqueue() – Add an element to the end of the queue
dequeue() – Remove an element from the front of the queue
display() – Display all elements in a queue
peek() – Get the value of the front of the queue without removing it
isfull() – Checks if the queue if full
isempty() – Checks if the queue is empty

5. **Reverse the given string using stack**

```
#include <stdio.h>
#define N 100
char stack[N];
int top=-1;
void push(char x){
  if(top==N-1){
     printf("Overflow");
  }
  else{
     top++;
     stack[top]=x;
  }
}
void pop(){
   if(top==-1){
      printf("Underflow");
   }
   else{
      printf("%c", stack[top]);
      top--;
   }
}
```

```
int main(){
   char string[N];
   printf("Enter the string to be reversed: ");
   scanf("%s", string);
   int size = sizeof(string)/sizeof(string[0]);
   for(int i=0;i<size;i++){
      push(string[i]);
   }
   printf("The reversed string: ");
   for(int i=0;i<size;i++){
      pop();
   }
}
```

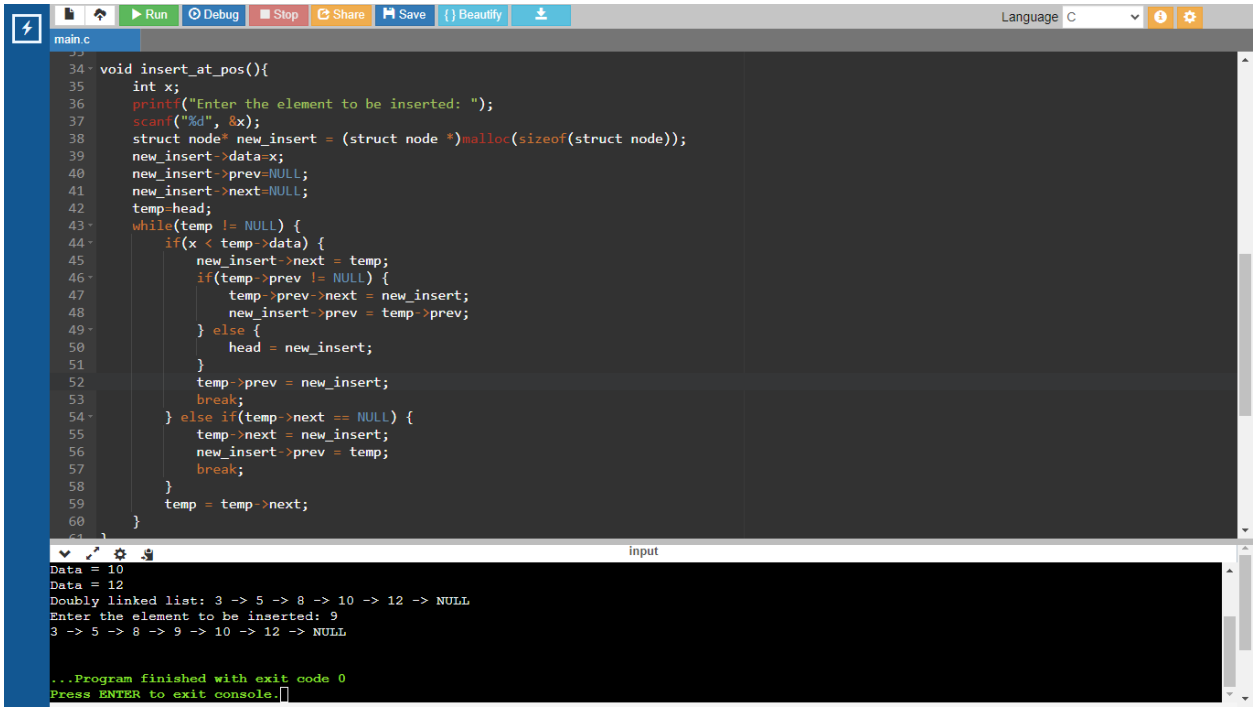


6. **Insert value in sorted way in a sorted doubly linked list. Given a sorted doubly linked list and a value to insert, write a function to insert the value in sorted way.**

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

struct node{
   int data;
   struct node* prev;
   struct node* next;
};

struct node *head, *temp, *newnode;

void create(){
   int n, i;
   printf("Number of nodes = ");
   scanf("%d", &n);
```

```
    printf("Enter the elements in a sorted way\n");
    for(i=0;i<n;i++){
        newnode = (struct node *)malloc(sizeof(struct node));
        printf("Data = ");
        scanf("%d", &newnode->data);
        newnode->prev = NULL;
        newnode->next = NULL;
        if(head==NULL){
            head=temp=newnode;
        }
        else{
            temp->next=newnode;
            newnode->prev=temp;
            temp=newnode;
        }
    }
}

void insert_at_pos(){
    int x;
    printf("Enter the element to be inserted: ");
    scanf("%d", &x);
    struct node* new_insert = (struct node *)malloc(sizeof(struct node));
    new_insert->data=x;
    new_insert->prev=NULL;
    new_insert->next=NULL;
    temp=head;
    while(temp != NULL) {
        if(x < temp->data) {
            new_insert->next = temp;
            if(temp->prev != NULL) {
                temp->prev->next = new_insert;
                new_insert->prev = temp->prev;
            } else {
                head = new_insert;
            }
            temp->prev = new_insert;
            break;
        } else if(temp->next == NULL) {
            temp->next = new_insert;
            new_insert->prev = temp;
            break;
        }
        temp = temp->next;
    }
}

void display(){
    temp=head;
```

```
    while(temp!=NULL){
        printf("%d -> ", temp->data);
        temp= temp->next;
    }
    printf("NULL\n");
}

int main(){
    create();
    printf("Doubly linked list: ");
    display();
    insert_at_pos();
    display();
}
```



Output: 8

7. **Write a C program to insert/delete and count the number of elements in a queue.**

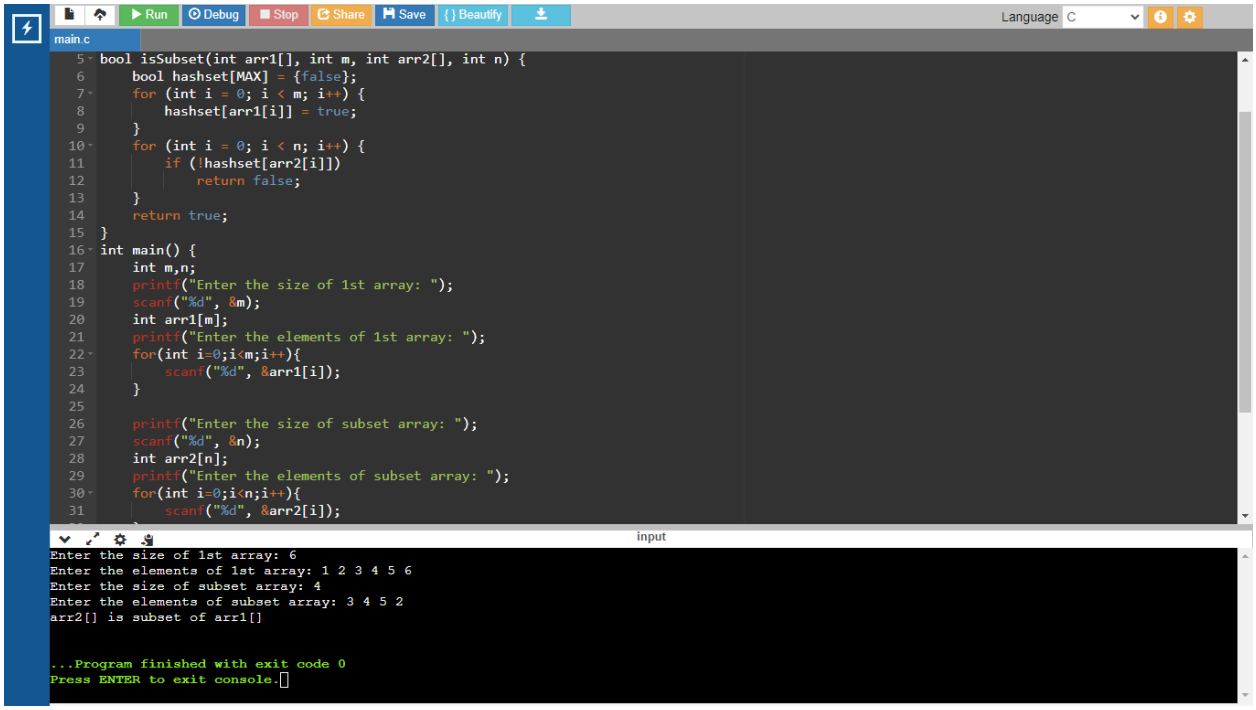
```
#include <stdio.h>
#define N 5
int queue[N];
int front = -1;
int rear = -1;

void enqueue(int x){
    if(rear==N-1){
        printf("Overflow\n");
    }
    else if((front==-1)&&(rear==-1)){
        front=rear=0;
        queue[rear]=x;
    }
    else{
```

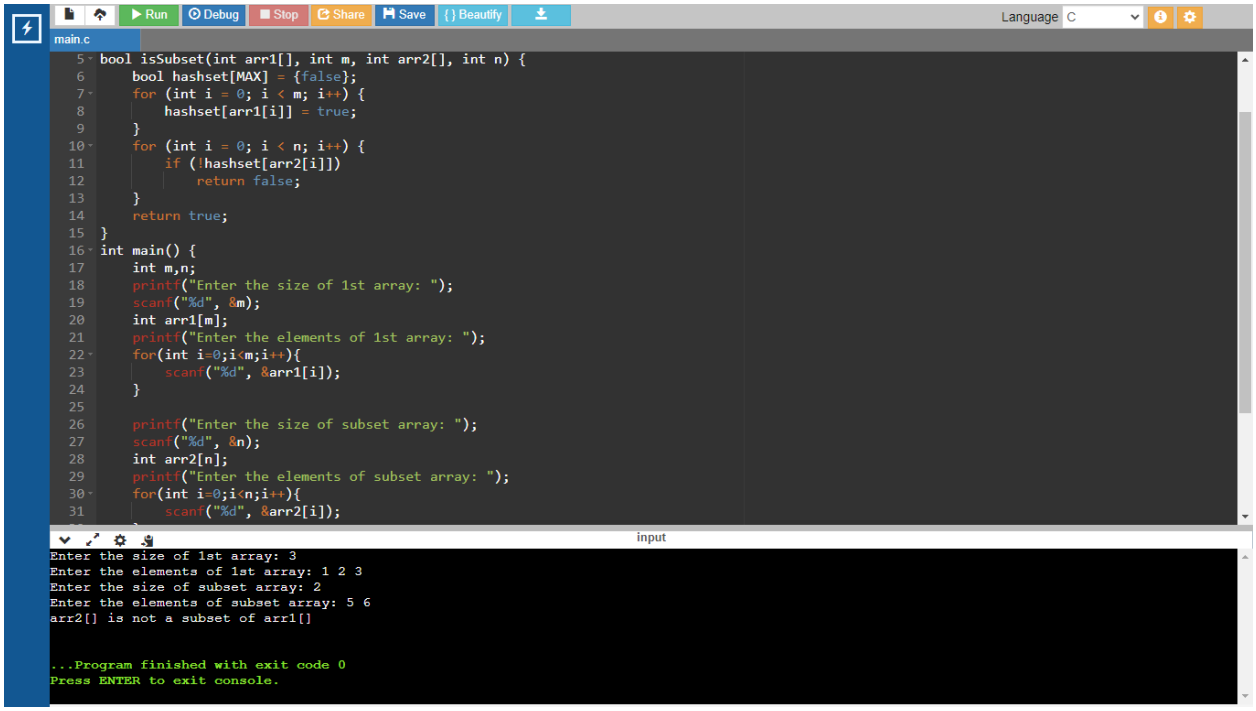
```
        rear++;
        queue[rear]=x;
    }
}

void dequeue(){
    if((front==-1)&&(rear==-1)){
        printf("\nUnderflow");
    }
    else if(front==rear){
        front=rear=-1;
    }
    else{
        front++;
    }
}

void display(){
    if((front==-1)&&(rear==-1)){
        printf("\nQueue is empty");
    }
    else{
        for(int i=front;i<rear+1;i++){
            printf("%d  ", queue[i]);
        }
    }
    printf("\n");
}

void count(){
    int countVar=0;
    if((front==-1)&&(rear==-1)){
        countVar=0;
    }
    else{
        for(int i=front;i<=rear;i++){
            countVar++;
        }
    }
    printf("Number of elements in the queue: %d\n", countVar);
}
void main(){
    printf("Initialize a queue!\n");
    count();
    printf("Insert some elements into the queue: \n");
    enqueue(1);
    enqueue(2);
    enqueue(3);
    printf("The queue elements are: \n");
```

```
    display();
    printf("Delete two elements from the said queue: \n");
    dequeue();
    dequeue();
    printf("The queue elements are: \n");
    display();
    count();
    printf("Insert another element into the queue: \n");
    enqueue(4);
    printf("The queue elements are: \n");
    display();
    count();
}
```

8. **Write a C program to Find whether an array is a subset of another array.**

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 100

bool isSubset(int arr1[], int m, int arr2[], int n) {
   bool hashset[MAX] = {false};
   for (int i = 0; i < m; i++) {
      hashset[arr1[i]] = true;
   }
   for (int i = 0; i < n; i++) {
      if (!hashset[arr2[i]])
         return false;
   }
   return true;
}
int main() {
   int m,n;
   printf("Enter the size of 1st array: ");
   scanf("%d", &m);
   int arr1[m];
   printf("Enter the elements of 1st array: ");
   for(int i=0;i<m;i++){
      scanf("%d", &arr1[i]);
   }

   printf("Enter the size of subset array: ");
   scanf("%d", &n);
   int arr2[n];
   printf("Enter the elements of subset array: ");
   for(int i=0;i<n;i++){
      scanf("%d", &arr2[i]);
   }

   if (isSubset(arr1, m, arr2, n))
      printf("arr2[] is subset of arr1[]\n");
   else
      printf("arr2[] is not a subset of arr1[]\n");

   return 0;
}
```


Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
bool isSubset(int arr1[], int m, int arr2[], int n) {
bool hashset[MAX] = {false};
for (int i = 0; i < m; i++) {
hashset[arr1[i]] = true;
}
for (int i = 0; i < n; i++) {
if (!hashset[arr2[i]])
return false;
}
return true;
}
int main() {
int m,n;
printf("Enter the size of 1st array: ");
scanf("%d", &m);
int arr1[m];
printf("Enter the elements of 1st array: ");
for(int i=0;i<m;i++){
scanf("%d", &arr1[i]);
}
printf("Enter the size of subset array: ");
scanf("%d", &n);
int arr2[n];
printf("Enter the elements of subset array: ");
for(int i=0;i<n;i++){
scanf("%d", &arr2[i]);
input
Enter the size of 1st array: 6
Enter the elements of 1st array: 1 2 3 4 5 6
Enter the size of subset array: 4
Enter the elements of subset array: 3 4 5 2
arr2[] is subset of arr1[]
...Program finished with exit code 0
Press ENTER to exit console.


Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
bool isSubset(int arr1[], int m, int arr2[], int n) {
bool hashset[MAX] = {false};
for (int i = 0; i < m; i++) {
hashset[arr1[i]] = true;
}
for (int i = 0; i < n; i++) {
if (!hashset[arr2[i]])
return false;
}
return true;
}
int main() {
int m,n;
printf("Enter the size of 1st array: ");
scanf("%d", &m);
int arr1[m];
printf("Enter the elements of 1st array: ");
for(int i=0;i<m;i++){
scanf("%d", &arr1[i]);
}
printf("Enter the size of subset array: ");
scanf("%d", &n);
int arr2[n];
printf("Enter the elements of subset array: ");
for(int i=0;i<n;i++){
scanf("%d", &arr2[i]);
input
Enter the size of 1st array: 3
Enter the elements of 1st array: 1 2 3
Enter the size of subset array: 2
Enter the elements of subset array: 5 6
arr2[] is not a subset of arr1[]
...Program finished with exit code 0
Press ENTER to exit console.