ADVANCED C PROGRAMMING ASSESSMENT MODULE 2

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

Input:

2 -> 3 -> 3 -> 4

Output:

2 -> 3 -> 4

Solution:

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


};
void removeDuplicates(struct Node* head)
{
   struct Node* current=head;
   struct Node* next_next;
   if (current==NULL)
     return;
   while(current->next!=NULL)
   {
    if(current->data==current->next->data)
    {
       next_next=current->next->next;
       free(current->next);
       current->next=next_next;
    }
    else
```

```
    {
        current=current->next;
    }
   }
}
void push(struct Node** head_ref,int new_data)
{
   struct Node* new_node=(struct Node*)malloc(sizeof(struct Node));
   new_node->data=new_data;
   new_node->next=(*head_ref);
   (*head_ref)=new_node;
}
void print(struct Node* node)
{
 while(node != NULL)
 {
    if(node->next==NULL)
    {
    printf("%d",node->data);
    break;
    }
    else
    {
    printf("%d->",node->data);
    node=node->next;
    }

 }
 printf("\n");
}
int main()
{
```

```
    struct Node* head =NULL;
    push(&head,4);
    push(&head,3);
    push(&head,3);
    push(&head,2);
    printf("Linked list before removing duplicate elements: \n");
    print(head);
    removeDuplicates(head);
    printf("Linked list after removing duplicate elements: \n");
    print(head);
    return 0;

}
```

OUTPUTS:

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

};
void removeDuplicates(struct Node* head)
{
    struct Node* current=head;
    struct Node* next_next;
    if (current==NULL)
       return;
    while(current->next!=NULL)
    {
     if(current->data==current->next->data)
     {
         next_next=current->next->next;
         free(current->next);
         current->next=next_next;
     }
     else
     {
         current=current->next;
     }
    }
```

```
input
Linked list before removing duplicate elements:
2->3->3->4
Linked list after removing duplicate elements:
2->3->4


...Program finished with exit code 0
Press ENTER to exit console.
```

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

Input: (When N=2)

a b c d e

Output:

c d e a b

Input: (When N=4)

a b c d e f g h

Output:

e f g h a b c d

Solution:

```
#include<stdio.h>
#include<stdlib.h>
#include <stdio.h>
struct node{
    char data;
    struct node *previous;
    struct node *next;
};
int size = 0;
struct node *head, *tail = NULL;
void add(char data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = data;
    if(head == NULL) {
        head = tail = newNode;
        head->previous = NULL;
        tail->next = NULL;
    }
    else {
        tail->next = newNode;
        newNode->previous = tail;
        tail = newNode;
        tail->next = NULL;
```

```
    }
    size++;
}
void rotateList(int n) {
    struct node *current = head;
    if(n == 0 || n >= size)
        return;
    else {
        for(int i = 1; i < n; i++)
            current = current->next;
        tail->next = head;
        head = current->next;
        head->previous = NULL;
        tail = current;
        tail->next = NULL;
}
}
void print() {
    struct node *current = head;
    while(current != NULL) {
        printf("%c ", current->data);
        current = current->next;
    }
    printf("\n");
}

int main()
{
    add('a');
```

```
    add('b');
    add('c');
    add('d');
    add('e');
    add('f');
    add('g');
    add('h');
    printf("List before rotating: \n");
    print();
    rotateList(4);
    printf("List after rotating: \n");
    print();
   return 0;
}
```

OUTPUTS:

```
/*****************************************************************************/
#include<stdio.h>
#include<stdlib.h>
#include <stdio.h>
struct node{
    char data;
    struct node *previous;
    struct node *next;
};
int size = 0;
struct node *head, *tail = NULL;
void add(char data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = data;
    if(head == NULL) {
        head = tail = newNode;
        head->previous = NULL;
        tail->next = NULL;
    }
    else {
        tail->next = newNode;
        newNode->previous = tail;
        tail = newNode;
        tail->next = NULL;
    }
```

```
List before rotating:
a b c d e f g h
List after rotating:
e f g h a b c d


...Program finished with exit code 0
Press ENTER to exit console.
```

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

Input

4 2 7 5 1

Output

1 2 4 5 7

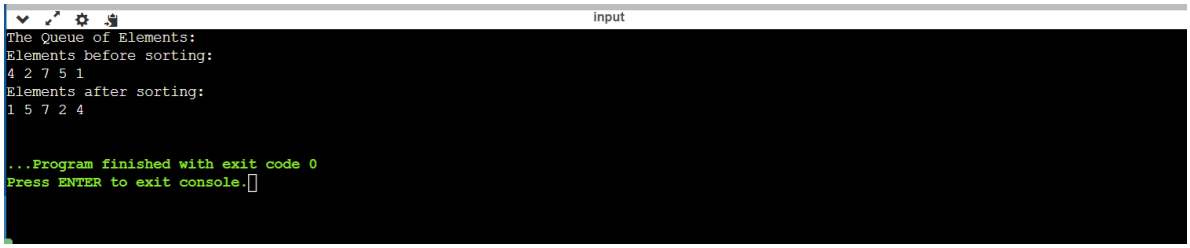
Solution:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
int queue[MAX];
int front=-1;
int back=-1;
void enqueue(int element)
{
 if(back==MAX-1)
 {
    printf("Queue is full\n");
 }
 if(front==-1)
 {
    front=0;
 }
 back++;
 queue[back]=element;
}
void Queue_asc_sort()
{
```

```
    int i,j,temp;
    int size=back-front+1;
    for(i=0;i<size-1;i++)
    {
        for(j=i+1;j<size;j++)
        {
            temp=queue[i];
            queue[i]=queue[j];
            queue[j]=temp;
        }
    }
}
void display()
{
    if(front==-1)
    {
        printf("Queue is empty");

    }
    else
    {
        for(int i=front;i<=back;i++)
        {
            printf("%d ",queue[i]);
        }
        printf("\n");
    }
}
int main()
```

```
{
    printf("The Queue of Elements:\n");
    enqueue(4);
    enqueue(2);
    enqueue(7);
    enqueue(5);
    enqueue(1);
    printf("Elements before sorting:\n");
    display();
    printf("Elements after sorting:\n");
    Queue_asc_sort();
    display();
    return 0;
}
```

OUTPUTS:



```
/*****************************************************************************/
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
int queue[MAX];
int front=-1;
int back=-1;
void enqueue(int element)
{
  if(back==MAX-1)
  {
      printf("Queue is full\n");
  }
  if(front==-1)
  {
      front=0;
  }
  back++;
  queue[back]=element;
}
void Queue_asc_sort()
{
    int i,j,temp;
```

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

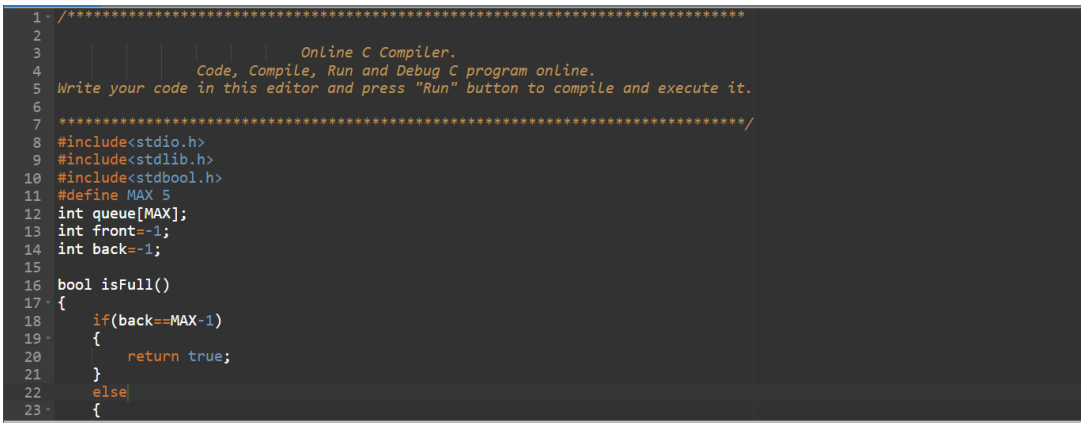
Solution:

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#define MAX 5
int queue[MAX];
int front=-1;
int back=-1;

bool isFull()
{
   if(back==MAX-1)
   {
      return true;
   }
   else
   {
      return false;
   }
}
bool isEmpty()
{
   if(front==-1 && back ==-1)
   {
      return true;
   }
   else
   {
      return false;
```

```
    }
}
void peak()
{
    if(front==-1)
    {
        printf("The Queue is empty\n");
    }
    else
    {
        printf("%d\n",queue[front]);
    }
}
void enqueue(int element)
{
  if(isFull())
  {
      printf("Queue is full\n");
  }
  if(front==-1)
  {
      front=0;
  }
  back++;
  queue[back]=element;
}

void dequeue()
{
```

```
    if(isEmpty())
    {
        printf("The Queue is Empty\n");
    }
    else
    {
        printf("The Popped element is %d\n",queue[front]);
        front++;
        if(front>back)
        {
            front=back=-1;
        }
    }
}
void display()
{
    if(front==-1)
    {
        printf("Queue is empty");

    }
    else
    {
        for(int i=front;i<=back;i++)
        {
            printf("%d ",queue[i]);
        }
        printf("\n");
    }
```

```
}
int main()
{
   printf("The Queue of Elements:\n");
   enqueue(4);
   enqueue(2);
   peak();
   enqueue(7);
   enqueue(5);
   enqueue(1);
   enqueue(3);
   display();
   dequeue();
   display();
   dequeue();
   dequeue();
   dequeue();
   dequeue();
   dequeue();
   isEmpty();
   return 0;
}
```



```
/******************************************************************************

                              Online C Compiler.
                  Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#define MAX 5
int queue[MAX];
int front=-1;
int back=-1;

bool isFull()
{
    if(back==MAX-1)
    {
        return true;
    }
    else
    {
```

5. Reverse the given string using stack

Input: (string)

"LetsLearn"

Output: (string)

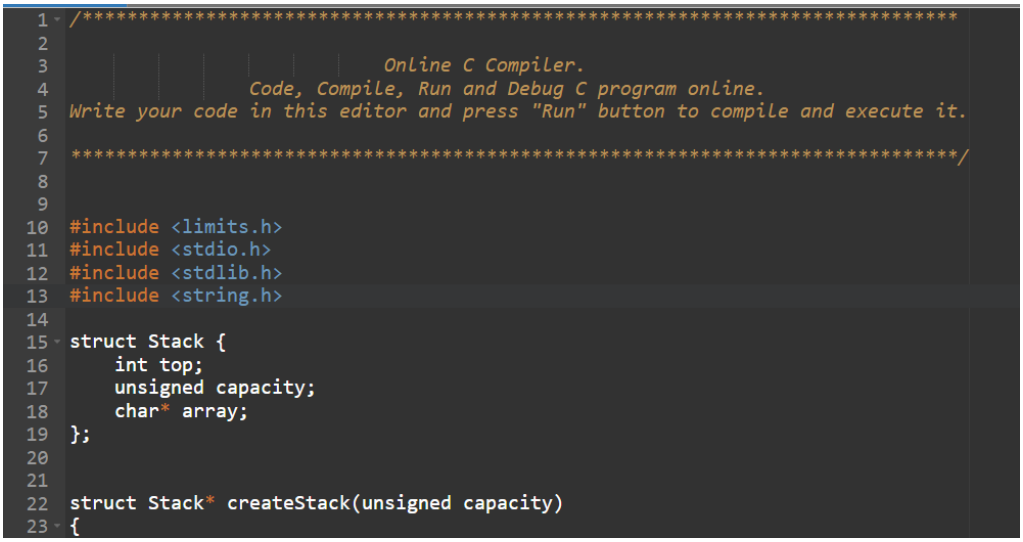
"nraeLsteL"

Solution:

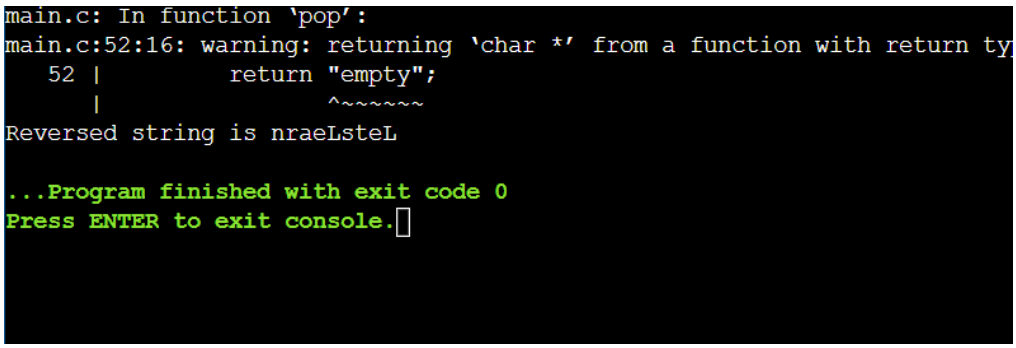
```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
 struct Stack {
    int top;
    unsigned capacity;
    char* array;
};
 struct Stack* createStack(unsigned capacity)
{
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
```

```
    stack->array = (char*)malloc(stack->capacity * sizeof(char));
    return stack;
}
 int isFull(struct Stack* stack)
{
    return stack->top == stack->capacity - 1;
}
int isEmpty(struct Stack* stack)
{
    return stack->top == -1;
}
 void push(struct Stack* stack, char item)
{
    if (isFull(stack))
        return;
    stack->array[++stack->top] = item;
}
 char pop(struct Stack* stack)
{
    if (isEmpty(stack))
        return "empty";
    return stack->array[stack->top--];
}
 void reverse(char str[])
{

    int n = strlen(str);
    struct Stack* stack = createStack(n);
```

```
    int i;
    for (i = 0; i < n; i++)
        push(stack, str[i]);

    for (i = 0; i < n; i++)
        str[i] = pop(stack);
}
 int main()
{
    char str[] = "LetsLearn";
    reverse(str);
    printf("Reversed string is %s", str);
    return 0;
}
```



```
/*******************************************************************************

                            Online C Compiler.
                    Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/


#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Stack {
    int top;
    unsigned capacity;
    char* array;
};


struct Stack* createStack(unsigned capacity)
{
```



```
main.c: In function 'pop':
main.c:52:16: warning: returning 'char *' from a function with return ty
   52 |         return "empty";
      |                ^~~~~~~
Reversed string is nraeLsteL

...Program finished with exit code 0
Press ENTER to exit console.
```

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.

Solution:

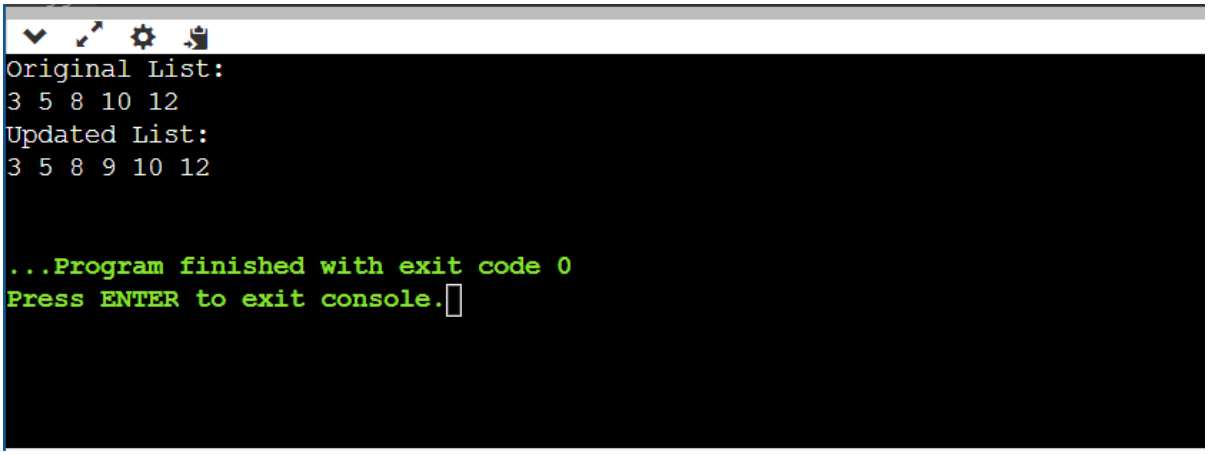
```
#include <stdio.h>

#include <stdlib.h>


struct Node {

    int value;

    struct Node *prev;

    struct Node *next;

};

struct Node* createNode(int value) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->value = value;

    newNode->prev = NULL;

    newNode->next = NULL;

    return newNode;

}

struct Node* insertNode(struct Node* head, int value) {

struct Node* newNode = createNode(value);

if (head == NULL) {

        return newNode;

    }


    if (value < head->value) {

        newNode->next = head;

        head->prev = newNode;

        return newNode;

    }
```

```
    struct Node* current = head;
    while (current->next != NULL && current->next->value < value) {
        current = current->next;
    }

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

    return head;
}

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

int main() {
    struct Node* head = createNode(3);
    struct Node* node1 = createNode(5);
    struct Node* node2 = createNode(8);
```

```
    struct Node* node3 = createNode(10);
    struct Node* node4 = createNode(12);
    head->next = node1;
    node1->prev = head;
    node1->next = node2;
    node2->prev = node1;
    node2->next = node3;
    node3->prev = node2;
    node3->next = node4;
    node4->prev = node3;
    printf("Original List:\n");
    printList(head);
    head = insertNode(head, 9);
    printf("Updated List:\n");
    printList(head);
    return 0;
}
```

main.c

```c
#include <stdlib.h>

struct Node {
    int value;
    struct Node *prev;
    struct Node *next;
};

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

struct Node* insertNode(struct Node* head, int value) {
    struct Node* newNode = createNode(value);

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



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

Expected Output:

Initialize a queue!
Check the queue is empty or not? Yes
Number of elements in queue: 0
Insert some elements into the queue:
Queue elements are: 1 2 3
Number of elements in queue: 3
Delete two elements from the said queue:
Queue elements are: 3
Number of elements in queue: 1
Insert another element into the queue:
Queue elements are: 3 4
Number of elements in the queue: 2

Solution:

```
#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#define MAX 5

int queue[MAX];

int front=-1;

int back=-1;


int count=0;

bool isFull()

{
```

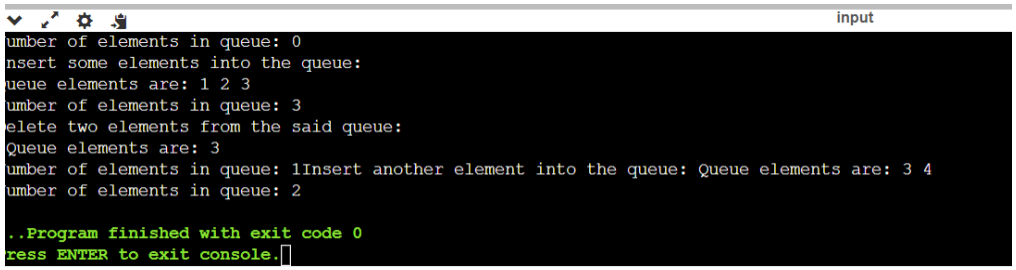
```
    if(back==MAX-1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
int emp=0;
void isEmpty()
{
    if(front==-1 && back ==-1)
    {
        printf("Yes");
        emp=0;
    }
    else
    {
        printf("No");
        emp=1;
    }
}


void peak()
{
    if(front==-1)
    {
```

```
        printf("The Queue is empty\n");
    }
    else
    {
        printf("%d\n",queue[front]);
    }
}
int f=0;
void enqueue(int element)
{
  if(isFull())
  {
     printf("Queue is full\n");
     f=0;
  }
  if(front==-1)
  {
     front=0;
  }
  back++;
  queue[back]=element;
  count++;
}

void dequeue()
{
    if(emp=0)
    {
        printf("The Queue is Empty\n");
```

```
    }
    else
    {
        front++;
        count--;
        if(front>back)
        {
            front=back=-1;
        }
    }
}
void display()
{
    if(front==-1)
    {
        printf("Queue is empty");

    }
    else
    {
        printf("Queue elements are: ");
        for(int i=front;i<=back;i++)
        {
            printf("%d ",queue[i]);
        }
        printf("\n");
    }
}
int main()
```

```
{

    printf("Initialize a queue!\n");
    printf("Check the queue is empty or not: ");
    isEmpty();
    printf("\n");
    printf("Number of elements in queue: %d\n",count);
    printf("Insert some elements into the queue:\n");
    enqueue(1);
    enqueue(2);
    enqueue(3);
    display();
    printf("Number of elements in queue: %d\n",count);
    printf("Delete two elements from the said queue:\n ");
    dequeue();
    dequeue();
    display();
    printf("Number of elements in queue: %d",count);
    printf("Insert another element into the queue: ");
    enqueue(4);
    display();
    printf("Number of elements in queue: %d",count);
    return 0;
}
```

```
    {
        return true;
    }
    else
    {
        return false;
    }
}
int emp=0;
void isEmpty()
{
    if(front==-1 && back ==-1)
    {
        printf("Yes");
        emp=0;
    }
    else
    {
        printf("No");
        emp=1;
    }
}
```



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

Input:

arr1[] = {11, 1, 13, 21, 3, 7}, arr2[] = {11, 3, 7, 1}

Output:

arr2[] is a subset of arr1[]

Input:

arr1[] = {10, 5, 2, 23, 19}, arr2[] = {19, 5, 3}

Output:

arr2[] is not a subset of arr1[]

```
#include<stdio.h>
#include<stdlib.h>
 int main()
{
   int arr1[]={11,1,13,21,3,7};
   int arr2[]={11,3,7,1};
   int arr3[]={10, 5, 2, 23, 19};
   int arr4[]={19,5,3};
```

```
int l=sizeof(arr1)/(arr1[0]);
int m=sizeof(arr2)/(arr2[0]);
int n=sizeof(arr3)/(arr3[0]);
int o=sizeof(arr4)/(arr4[0]);
int flag=0;
int f=0;
for(int i=0;i<m;i++)
{
   for(int j=0;j<l;j++)
   {
      if(arr2[i]==arr1[i])
      {
         flag=1;


      }
      else
      {
         break;
      }
   }
}
for(int i=0;i<o;i++)
{
   for(int j=0;j<n;j++)
   {
      if(arr4[i]==arr3[i])
      {
         f=1;
```

```
            }
            else
            {
                break;
            }
        }
    }

    if(flag==1)
    {

        printf("arr2[] is a subset of arr1[]\n");
    }
    else
    {
        printf("arr2[] is not a subset of arr1[]\n");
    }
     if(f==1)
    {
        printf("arr4[] is a subset of arr3[]\n");

    }
    else
    {
        printf("arr4[] is not a subset of arr3[]\n");
    }
    return 0;
}
```

OUTPUTS:

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


int main()
{
    int arr1[]={11,1,13,21,3,7};
    int arr2[]={11,3,7,1};
    int arr3[]={10, 5, 2, 23, 19};
    int arr4[]={19,5,3};
    int l=sizeof(arr1)/(arr1[0]);
    int m=sizeof(arr2)/(arr2[0]);
    int n=sizeof(arr3)/(arr3[0]);
    int o=sizeof(arr4)/(arr4[0]);
    int flag=0;
    int f=0;
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<l;j++)
        {
            if(arr2[i]==arr1[i])
            {
                flag=1;
```

```
arr2[] is a subset of arr1[]
arr4[] is not a subset of arr3[]


...Program finished with exit code 0
Press ENTER to exit console.
```