**Problem Statement:**

Design a menu-driven program that allows users to perform various operations on a **stack** implemented using an array. The program should provide options to push elements onto the stack, pop elements from the stack, check if the stack is empty or full, and display the elements of the stack. The user should be able to choose any of these operations from a menu and provide the required inputs.

The program should implement the following functionalities:

1. Stack Initialization:

2. Push:

3. Pop:

4. Check Empty:

5. Check Full:

6. Display:

7. Exit:

The program should display a menu with the above options and allow the user to select an operation by entering the corresponding menu number. After executing the selected operation, the program should return to the menu and continue until the user chooses the exit option.

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

#define n 10

struct stack

{

int stk[n];

int top;

};

typedef struct stack ST;

ST s;

int push()

{

if(s.top == n-1)

{

printf("/nStack Overflow : ");

}

else

{

int x;

printf("\nEnter an element to push : ");

scanf("%d", &x);

s.top++;

s.stk[s.top] = x;

}

}

int pop()

{

if(s.top == -1)

{

printf("\nStack Underflow");

}

else

{

s.top--;

}

}

int peek()

{

if(s.top == -1)

{

printf("\nStack Underflow");

}

else

{

printf("\n Peek element is : %d ", s.stk[s.top]);

}

}

void display()

{

int temp = s.top;

if(temp == -1)

{

printf("\nStack Underflow");

}

else

{

printf("\n");

while(temp>=0)

{

printf("\n %d ",s.stk[temp] );

temp--;

}

}

}

int main()

{

printf("Yogesh Pal Parmar MCA 2B 70");

s.top = -1;

int ch;

do

{

printf("\nEnter a Choice : 0: EXIT! / 1: Push : / 2: Pop : / 3: Peek : / 4: Display :\n");

scanf("%d", &ch);

switch(ch)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

default:

printf("\n Invalid choice! ");

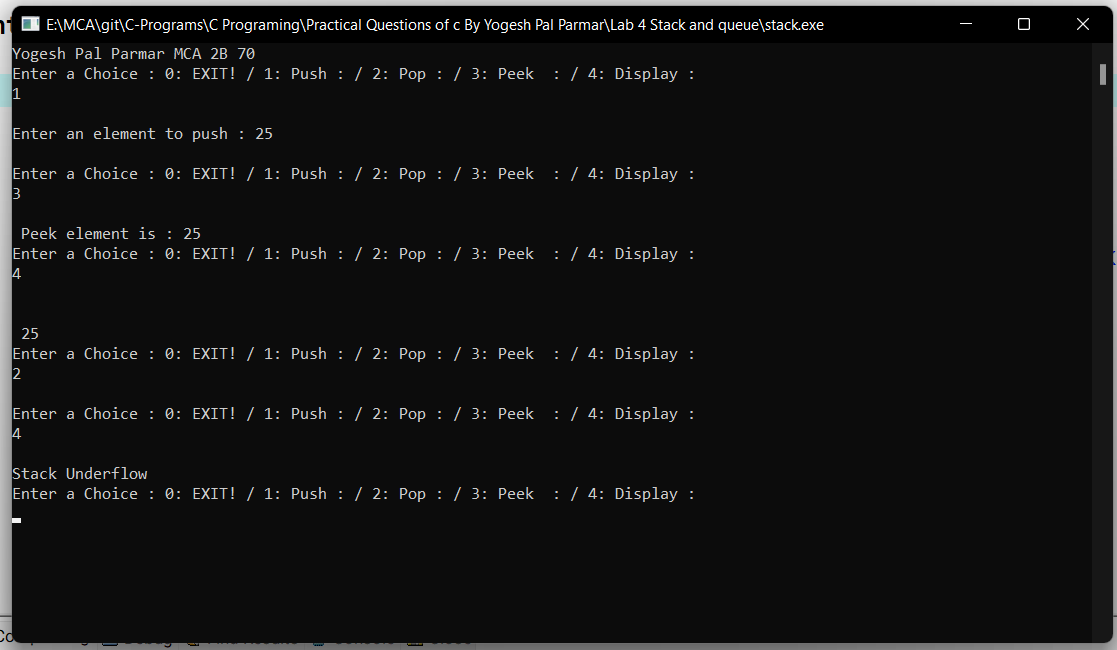
}

}while(ch != 0);

return 0;

}

**Output:**



**Problem Statement:**

Design a menu-driven program that allows users to perform various operations on a **queue** implemented using an array. The program should provide options to enqueue elements into the queue, dequeue elements from the queue, check if the queue is empty or full, and display the elements of the queue. The user should be able to choose any of these operations from a menu and provide the required inputs.

The program should implement the following functionalities:

1. Queue Initialization:

2. Enqueue:

3. Dequeue:

4. Check Empty:

5. Check Full:

6. Display:

7. Exit:

The program should display a menu with the above options and allow the user to select an operation by entering the corresponding menu number. After executing the selected operation, the program should return to the menu and continue until the user chooses the exit option.

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

#define MAX 10

int queue\_arr[MAX];

int rear=-1;

int front=-1;

void insert(int item);

int del();

int peek();

void display();

int isFull();

int isEmpty();

int main()

{

printf("Yogesh Pal Parmar MCA 2B 70\n");

int choice,item;

while(1)

{

printf("\n1.Insert\n");

printf("2.Delete\n");

printf("3.Display element at the front\n");

printf("4.Display all elements of the queue\n");

printf("5.Quit\n");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nInput the element for adding in queue : ");

scanf("%d",&item);

insert(item);

break;

case 2:

item=del();

printf("\nDeleted element is %d\n",item);

break;

case 3:

printf("\nElement at the front is %d\n",peek());

break;

case 4:

display();

break;

case 5:

exit(1);

default:

printf("\nWrong choice\n");

}/\*End of switch\*/

}/\*End of while\*/

return 0;

}/\*End of main()\*/

void insert(int item)

{

if( isFull() )

{

printf("\nQueue Overflow\n");

return;

}

if( front == -1 )

front=0;

rear=rear+1;

queue\_arr[rear]=item ;

}/\*End of insert()\*/

int del()

{

int item;

if( isEmpty() )

{

printf("\nQueue Underflow\n");

exit(1);

}

item=queue\_arr[front];

front=front+1;

return item;

}/\*End of del()\*/

int peek()

{

if( isEmpty() )

{

printf("\nQueue Underflow\n");

exit(1);

}

return queue\_arr[front];

}/\*End of peek()\*/

int isEmpty()

{

if( front==-1 || front==rear+1 )

return 1;

else

return 0;

}/\*End of isEmpty()\*/

int isFull()

{

if( rear==MAX-1 )

return 1;

else

return 0;

}/\*End of isFull()\*/

void display()

{

int i;

if ( isEmpty() )

{

printf("\nQueue is empty\n");

return;

}

printf("\nQueue is :\n\n");

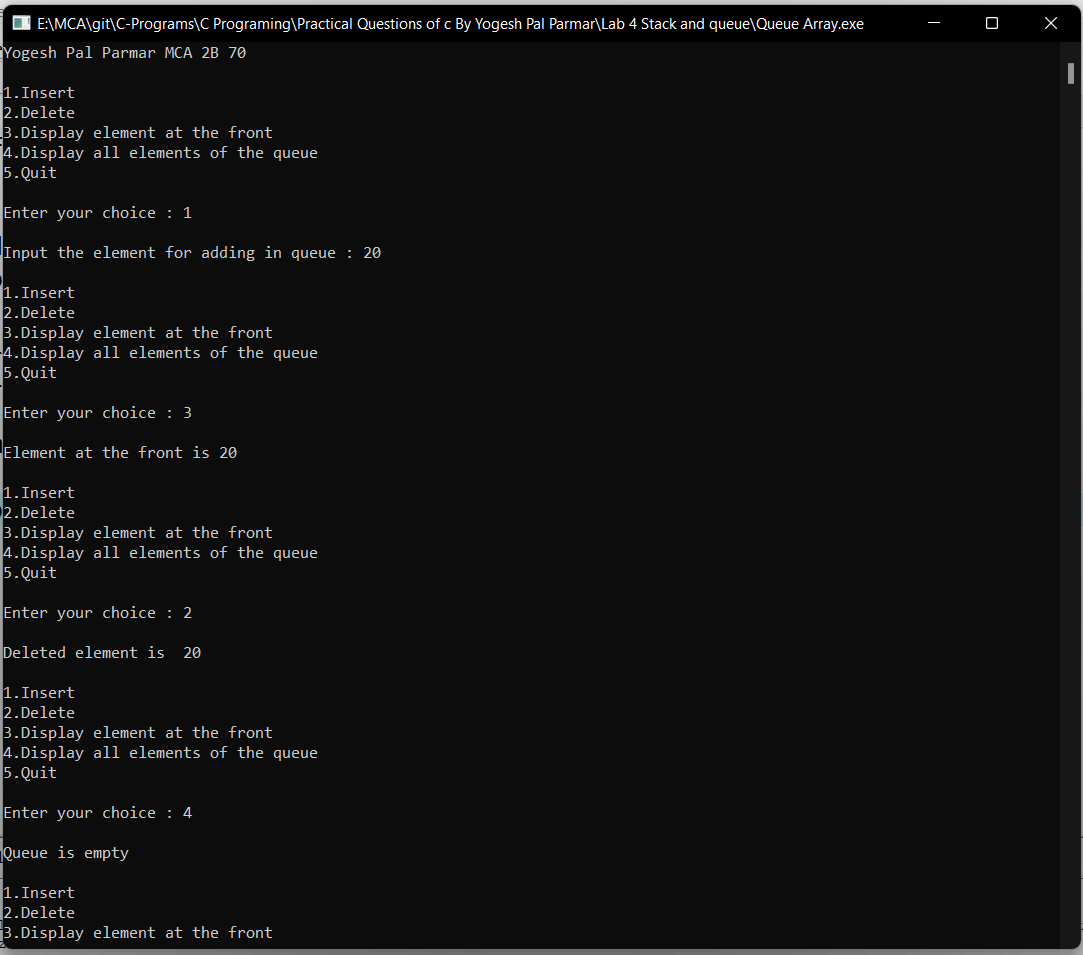
for(i=front;i<=rear;i++)

printf("%d ",queue\_arr[i]);

printf("\n\n");

}

**Output:**



**Problem Statement:**

Design a menu-driven program in C that allows users to perform various operations on a **stack** implemented using a linked list. The program should provide options to push elements onto the stack, pop elements from the stack, check if the stack is empty, display the elements of the stack, and exit the program. The user should be able to choose any of these operations from a menu and provide the required inputs.

The program should implement the following functionalities:

1. Stack Initialization:

2. Push Operation:

3. Pop Operation:

4. Check Empty:

5. Display Stack:

6. Exit:

The program should display a menu with the above options and allow the user to select an operation by entering the corresponding menu number. After executing the selected operation, the program should return to the menu and continue until the user chooses the exit option.

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

// Node structure for the linked list

struct Node {

int data;

struct Node\* next;

};

// Function to create a new node

struct Node\* createNode(int value) {

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

if (newNode == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

newNode->data = value;

newNode->next = NULL;

return newNode;

}

// Stack structure

struct Stack {

struct Node\* top;

};

// Function to initialize the stack

void initializeStack(struct Stack\* stack) {

stack->top = NULL;

}

// Function to check if the stack is empty

int isEmpty(struct Stack\* stack) {

return stack->top == NULL;

}

// Function to push an element onto the stack

void push(struct Stack\* stack, int value) {

struct Node\* newNode = createNode(value);

newNode->next = stack->top;

stack->top = newNode;

printf("%d pushed to the stack.\n", value);

}

// Function to pop an element from the stack

int pop(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Stack is empty. Cannot perform pop operation.\n");

return -1;

}

struct Node\* temp = stack->top;

int poppedValue = temp->data;

stack->top = stack->top->next;

free(temp);

return poppedValue;

}

// Function to display the elements of the stack

void display(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Stack is empty.\n");

return;

}

struct Node\* temp = stack->top;

printf("Stack elements: ");

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

int main() {

printf("Yogesh Pal Parmar MCA 2B 70");

struct Stack stack;

initializeStack(&stack);

int choice, value;

while (1) {

printf("\nStack Operations:\n");

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the value to push: ");

scanf("%d", &value);

push(&stack, value);

break;

case 2:

value = pop(&stack);

if (value != -1)

printf("Popped value: %d\n", value);

break;

case 3:

display(&stack);

break;

case 4:

printf("Exiting program.\n");

exit(0);

default:

printf("Invalid choice. Please try again.\n");

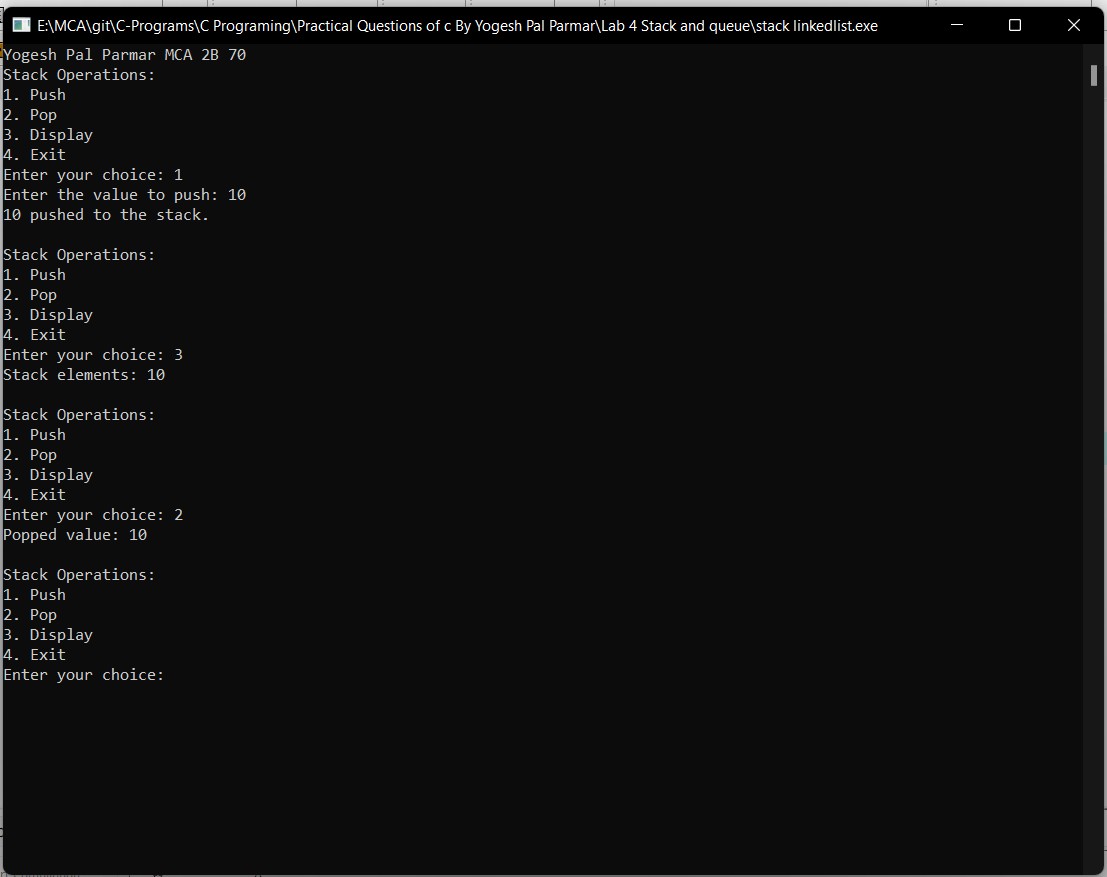
}

}

return 0;

}

**Output:**



**Problem Statement:**

Design a menu-driven program in C that allows users to perform various operations on a **queue** implemented using a linked list. The program should provide options to enqueue elements into the queue, dequeue elements from the queue, check if the queue is empty, display the elements of the queue, and exit the program. The user should be able to choose any of these operations from a menu and provide the required inputs.

The program should implement the following functionalities:

1. Queue Initialization:

2. Enqueue Operation:

3. Dequeue Operation:

4. Check Empty:

5. Display Queue:

6. Exit:

The program should display a menu with the above options and allow the user to select an operation by entering the corresponding menu number. After executing the selected operation, the program should return to the menu and continue until the user chooses the exit option.

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

// Node structure for the linked list

struct Node {

int data;

struct Node\* next;

};

// Queue structure

struct Queue {

struct Node\* front;

struct Node\* rear;

};

// Function to create a new node

struct Node\* createNode(int value) {

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

if (newNode == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

newNode->data = value;

newNode->next = NULL;

return newNode;

}

// Function to initialize the queue

void initializeQueue(struct Queue\* queue) {

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

}

// Function to check if the queue is empty

int isEmpty(struct Queue\* queue) {

return queue->front == NULL;

}

// Function to enqueue an element into the queue

void enqueue(struct Queue\* queue, int value) {

struct Node\* newNode = createNode(value);

if (isEmpty(queue)) {

queue->front = queue->rear = newNode;

} else {

queue->rear->next = newNode;

queue->rear = newNode;

}

printf("%d enqueued to the queue.\n", value);

}

// Function to dequeue an element from the queue

int dequeue(struct Queue\* queue) {

if (isEmpty(queue)) {

printf("Queue is empty. Cannot perform dequeue operation.\n");

return -1;

}

struct Node\* temp = queue->front;

int dequeuedValue = temp->data;

queue->front = queue->front->next;

if (queue->front == NULL) {

queue->rear = NULL;

}

free(temp);

return dequeuedValue;

}

// Function to display the elements of the queue

void display(struct Queue\* queue) {

if (isEmpty(queue)) {

printf("Queue is empty.\n");

return;

}

struct Node\* temp = queue->front;

printf("Queue elements: ");

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

int main() {

printf("Yogesh Pal Parmar MCA 2B 70");

struct Queue queue;

initializeQueue(&queue);

int choice, value;

while (1) {

printf("\nQueue Operations:\n");

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the value to enqueue: ");

scanf("%d", &value);

enqueue(&queue, value);

break;

case 2:

value = dequeue(&queue);

if (value != -1)

printf("Dequeued value: %d\n", value);

break;

case 3:

display(&queue);

break;

case 4:

printf("Exiting program.\n");

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

**Output:**

