B.M.S. COLLEGE OF ENGINEERING

Bull Temple Road, Basavanagudi, Bengaluru-590019, Karnataka.

LAB REPORT
on

Data Structures using C Lab

(23CS3PCDST)

Submitted by

Sinchana Hemanth (1BM23CS330)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Data Structures using C Lab (23CS3PCDST) carried out by **Sinchana Hemanth (1BM23CS330),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of Data Structures using C Lab (23CS3PCDST) work prescribed for the said degree.

Prasad GR
Assistant Professor
Department of CSE, BMSCE
Dr. Kavitha Sooda
Professor & HOD
Department of CSE, BMSCE

Index

Github Link: https://github.com/SinchanaHemanth/1BM23CS330-SinchanaHemanth.git

Program 1

Write a program to simulate the working of stack using an array with the following:

a) Push

b) Pop

c) Display

The program should print appropriate messages for stack overflow, stack underflow.

• Implement stack and ops using arrays

```
#include <stdlib.h>
#include <stdio.h>
int top, size;

void init()
{
    printf("Enter size of stack\n");
    scanf("%d", &size);
    top = -1;
}

bool isEmpty()
{
   return top==-1;
}

bool isFull()
{
   return top==size-1;
}

void push (int arr[], int x)
{
   if (!isFull())
   {
      top++;
      arr [top] = x;
      printf("Pushed %d to stack \n", x);
   }
   else
```

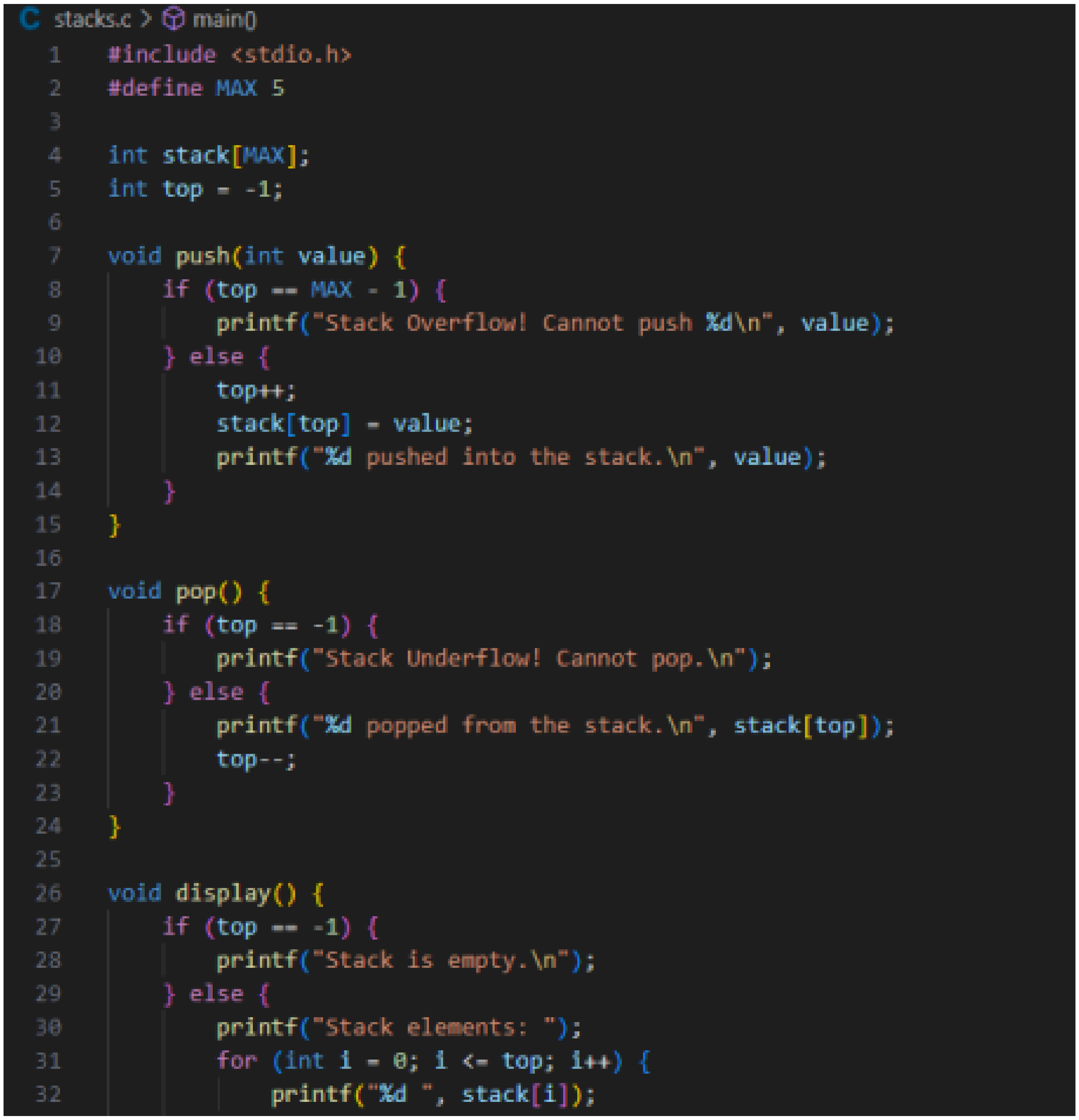
```
        {
          printf("Overflow\n");
        }
}
int pop(int arr[])
{
   if(isEmpty())
   {
      printf("Underflow\n");
      return 0;
   }
   else
   {
      int temp = arr[top];
      top--;
      return temp;
   }
}
int peek(int arr[])
{
   if(isEmpty())
   {
      printf("Stack is empty\n");
      return 0;
   }
   else
   {
      return arr[top];
   }
}
int main()
{
```

```
init();
int arr[size];

for (int i=0; i<size; i++)
{
    int element;
    printf("Enter element %d: ", i+1);
    scanf("%d", &element);
    push(arr, element);
}

for (int i=0; i<size; i++)
{
    printf("Popped : %d\n", pop(arr));
}
return 0;
}
```

Seen
[illegible]
1/10/24

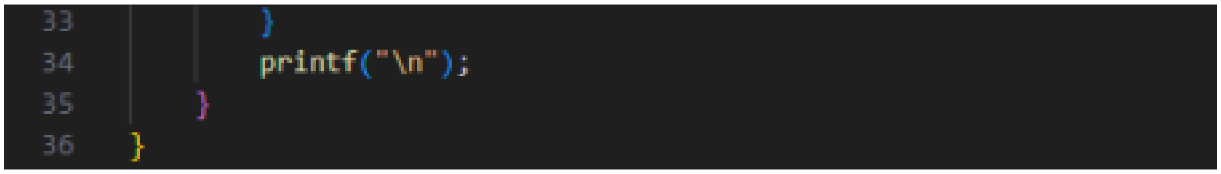
Output

```
Enter size of stack :
5
Enter element 1: 6
Pushed 6 to stack
Enter element 2: 7
Pushed 7 to stack
Enter element 3: 8
Pushed 8 to stack
Enter element 4: 9
Pushed 9 to stack
Enter element 5: 10
Pushed 10 to stack
Popped : 10
Popped : 9
Popped : 8
Popped : 7        Popped : 6
```

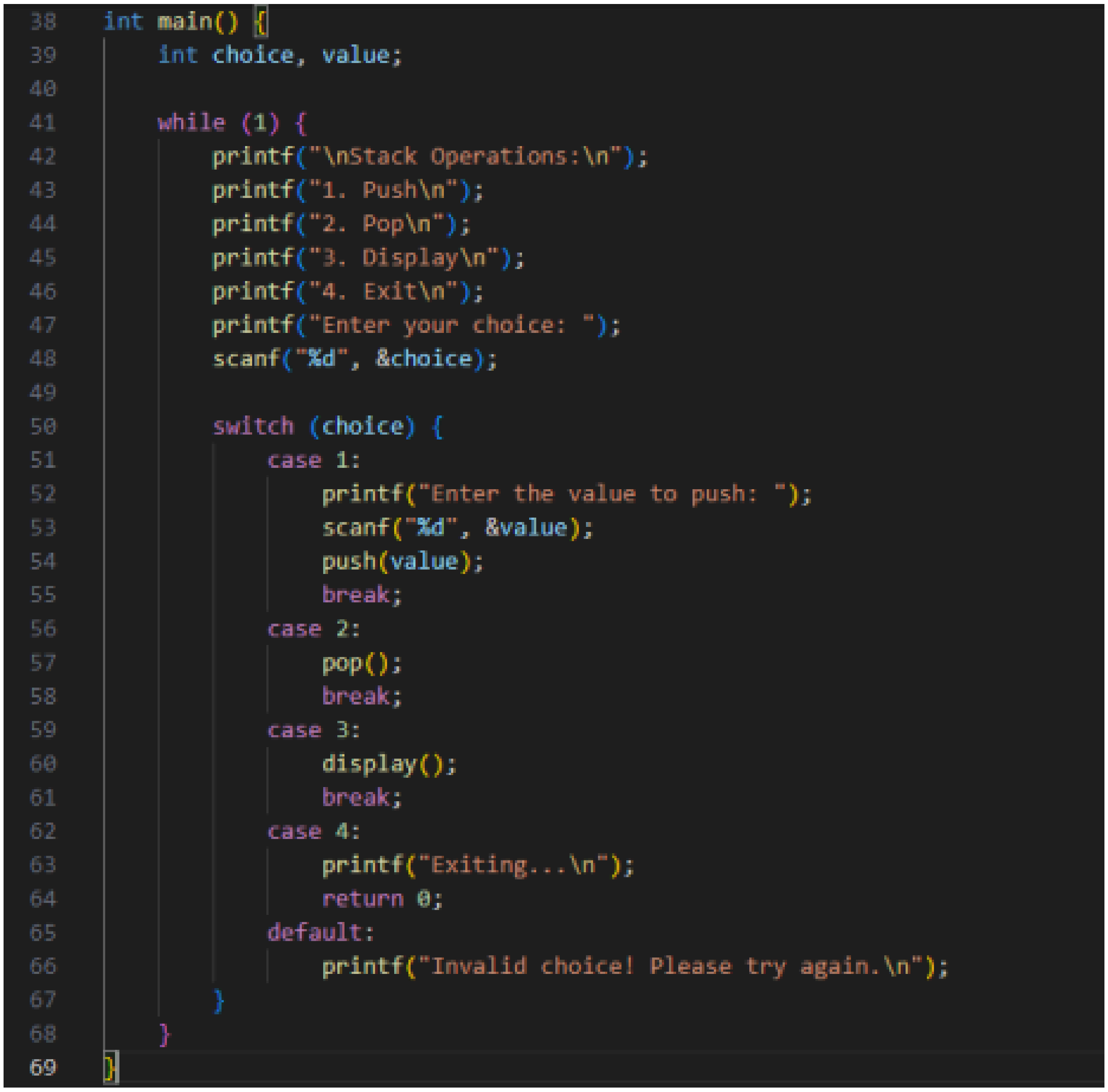


stacks.c > main()

```c
#include <stdio.h>
#define MAX 5

int stack[MAX];
int top = -1;

void push(int value) {
    if (top == MAX - 1) {
        printf("Stack Overflow! Cannot push %d\n", value);
    } else {
        top++;
        stack[top] = value;
        printf("%d pushed into the stack.\n", value);
    }
}

void pop() {
    if (top == -1) {
        printf("Stack Underflow! Cannot pop.\n");
    } else {
        printf("%d popped from the stack.\n", stack[top]);
        top--;
    }
}

void display() {
    if (top == -1) {
        printf("Stack is empty.\n");
    } else {
        printf("Stack elements: ");
        for (int i = 0; i <= top; i++) {
            printf("%d ", stack[i]);
```

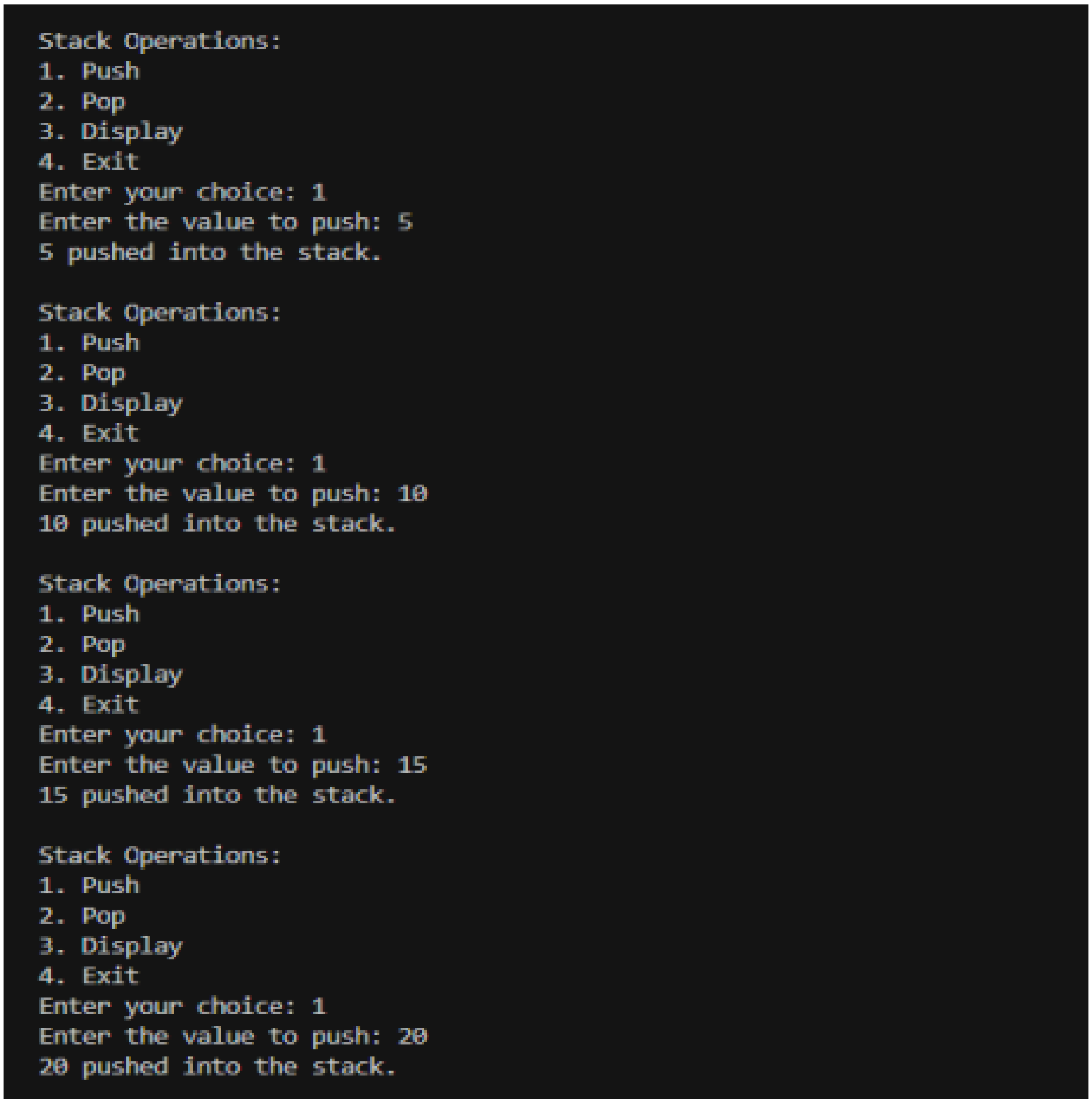


```c
        }
        printf("\n");
    }
}
```

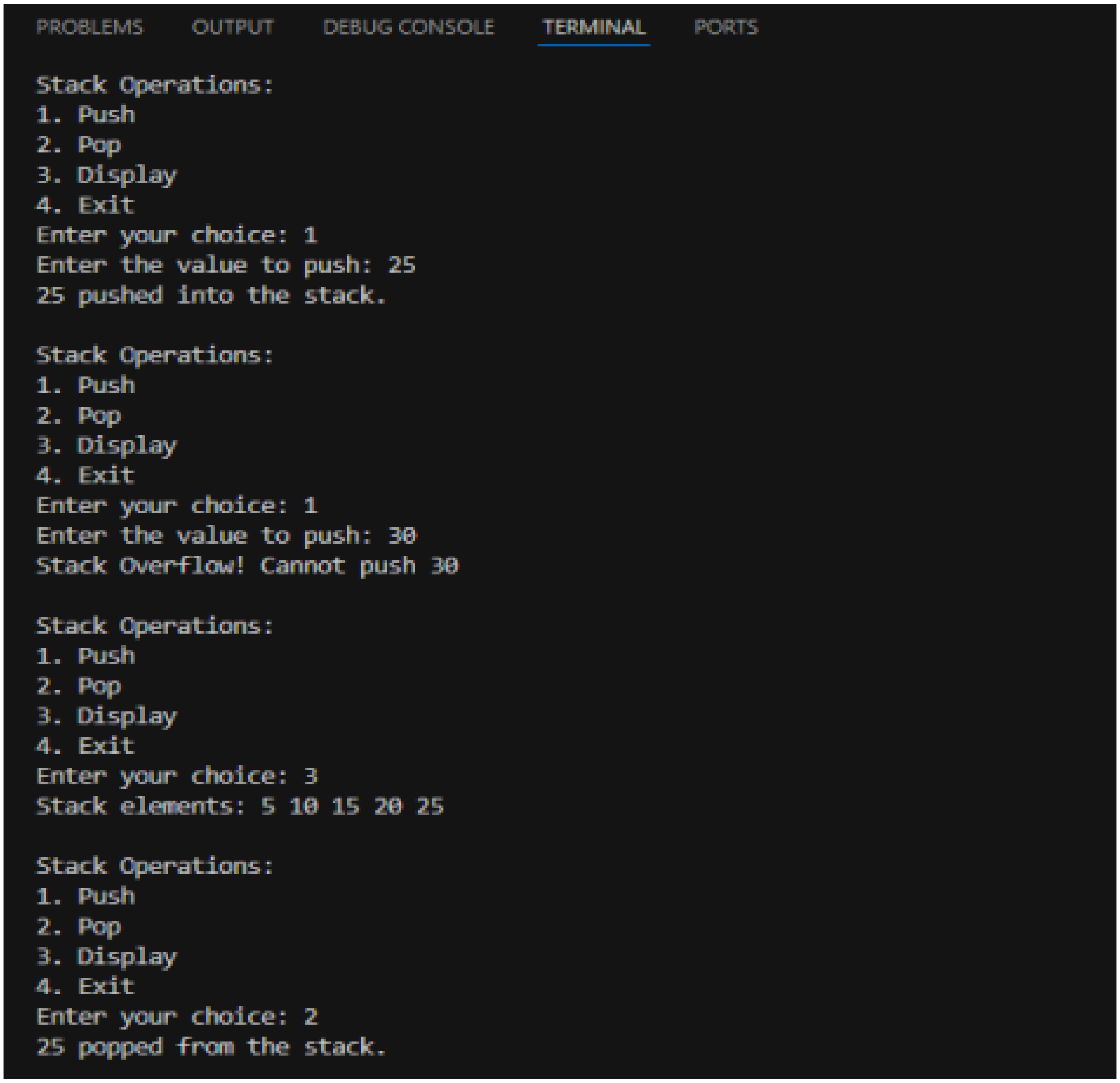


```c
int main() {
    int choice, value;

    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(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
}
```

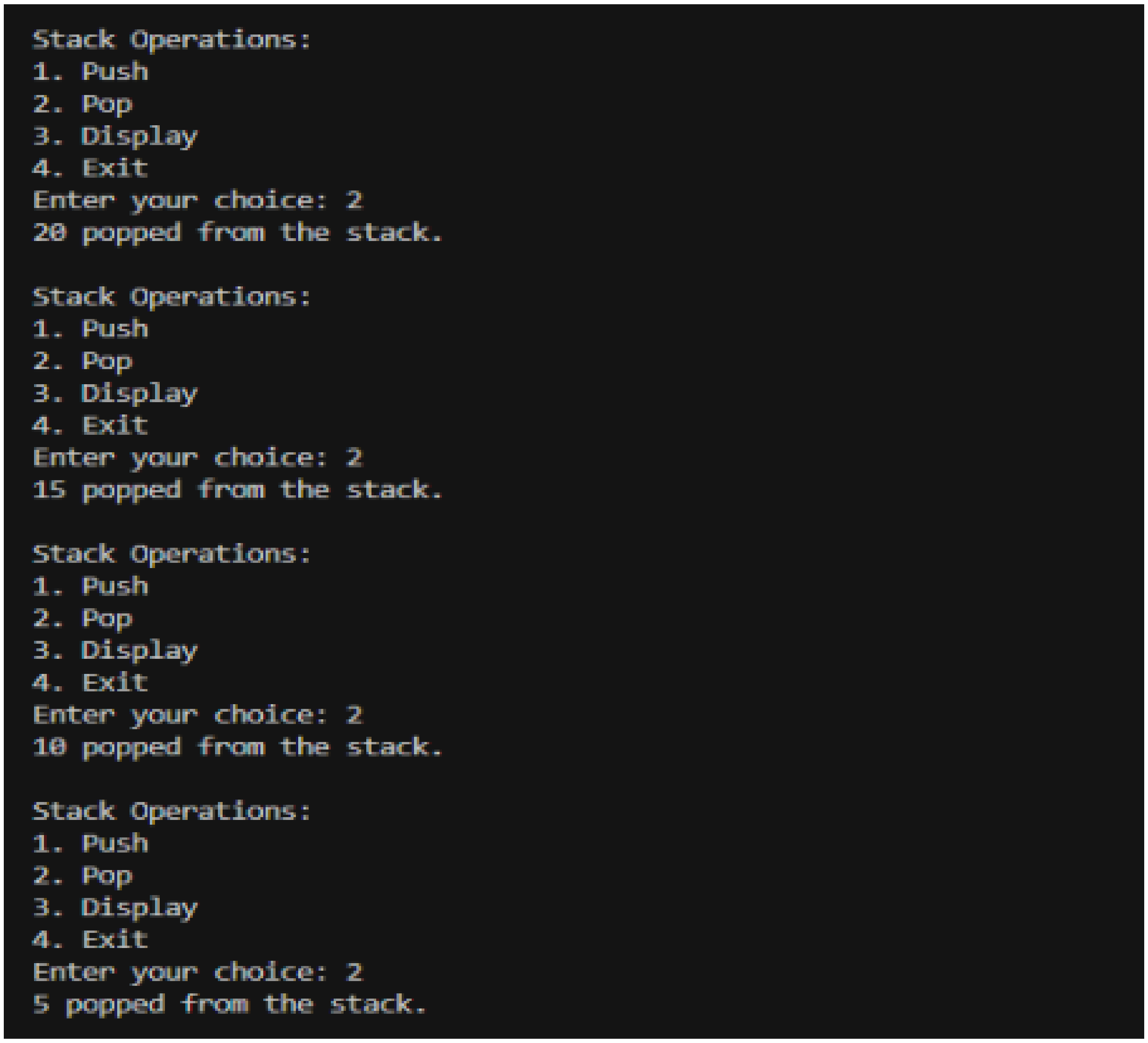
OUTPUT:



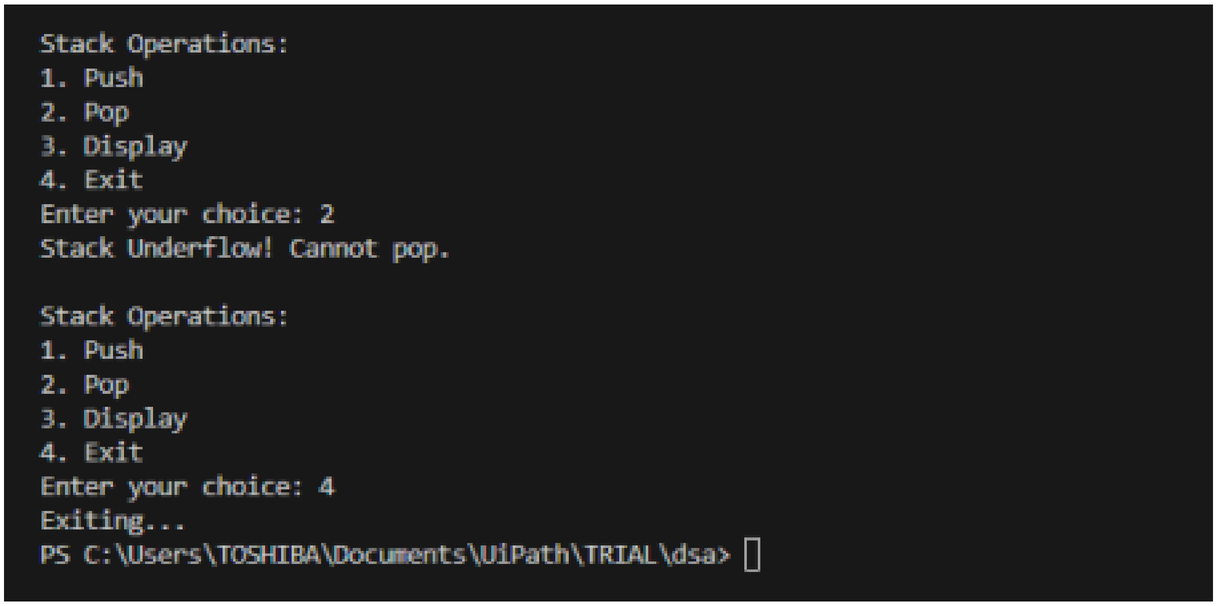
```
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push: 5
5 pushed into the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push: 10
10 pushed into the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push: 15
15 pushed into the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push: 20
20 pushed into the stack.
```



```
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push: 25
25 pushed into the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to push: 30
Stack Overflow! Cannot push 30

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack elements: 5 10 15 20 25

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
25 popped from the stack.
```



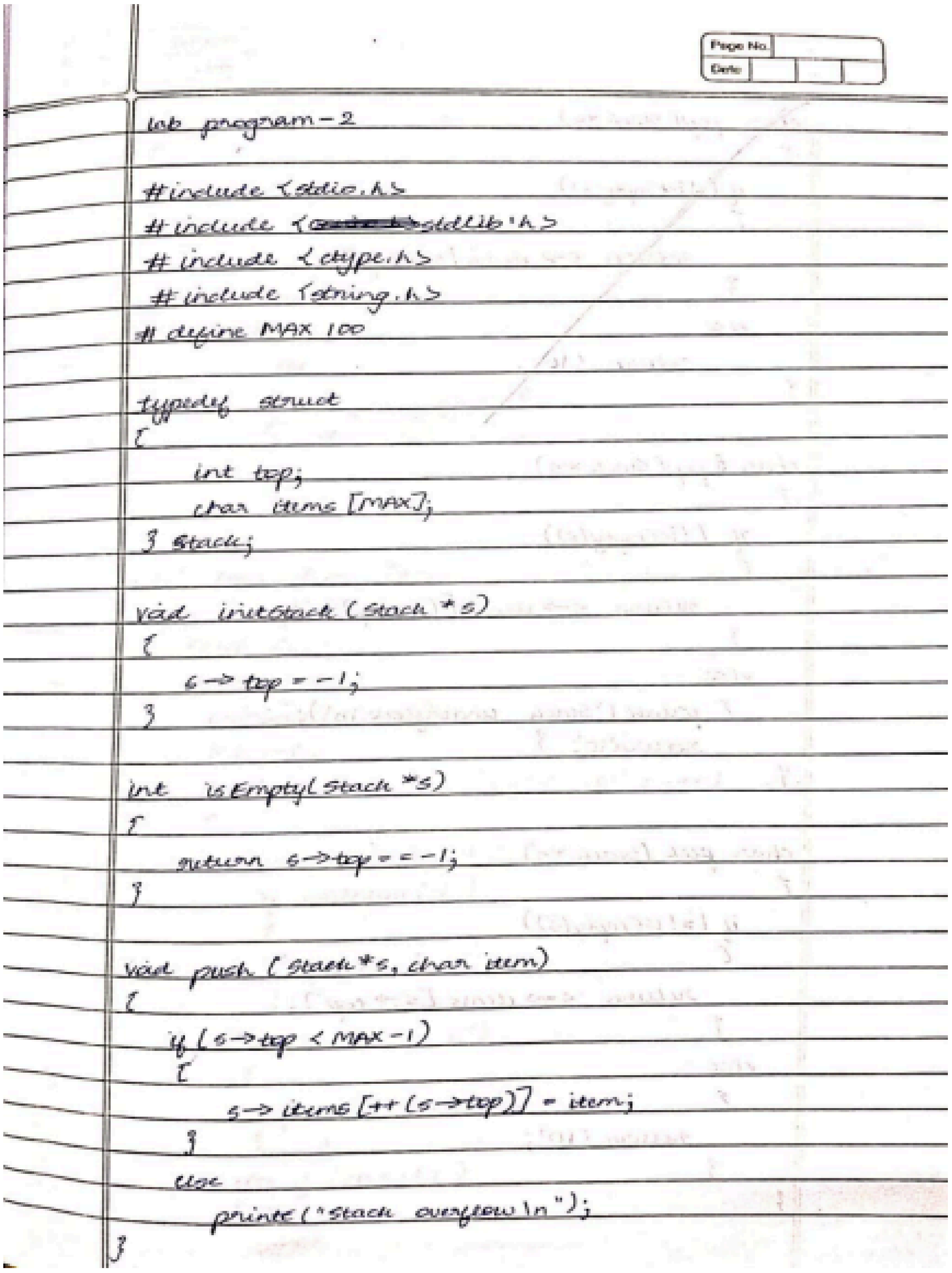
```
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
20 popped from the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
15 popped from the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
10 popped from the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
5 popped from the stack.
```



```
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Stack Underflow! Cannot pop.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 4
Exiting...
PS C:\Users\TOSHIBA\Documents\UiPath\TRIAL\dsa>
```

Program 2

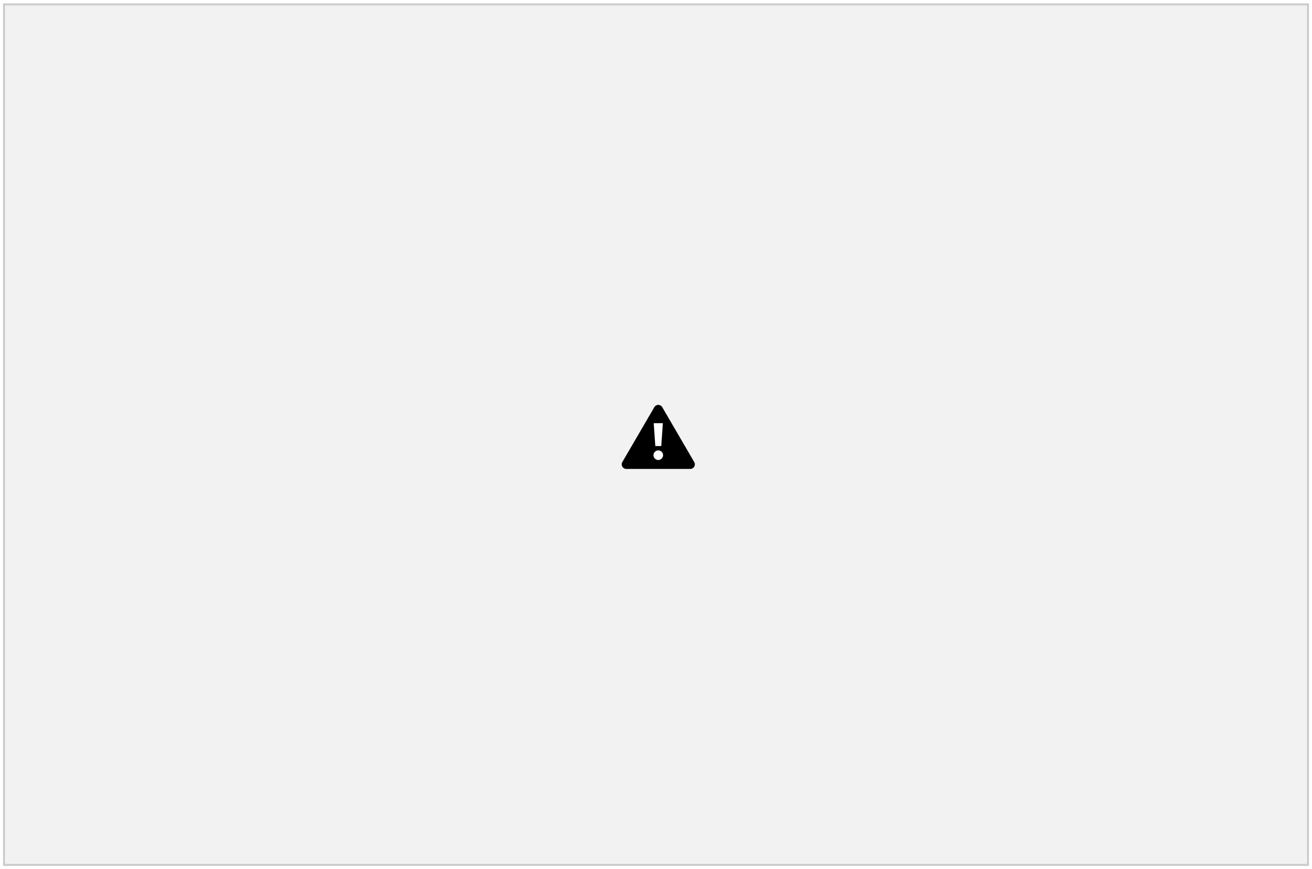
Write a program to convert given valid parenthesized infix arithmetic expressions to postfix expressions. The expression consists of single character operands and the binary operators are + (plus), - (minus), * (multiply), / (divide) and ^ (exponential).



```
lab program-2

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#define MAX 100

typedef struct
{
    int top;
    char items [MAX];
} stack;

void initStack (stack *s)
{
    s->top = -1;
}

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

void push (stack *s, char item)
{
    if (s->top < MAX -1)
    {
        s-> items [++ (s->top)] = item;
    }
    else
        printf ("stack overflow\n");
}
```

```
char pop(Stack *s)
{
    if (!isEmpty(s))
    {
        return s->items[(s->top)--];
    }
    else
     { printf("Stack underflow \n");
        return '\0'; }
}

char peek (Stack *s)
{
    if (!isEmpty(s))
     {
        return s->items[s->top];
     }
   else
     {
        return '\0';
     }
}
```

19

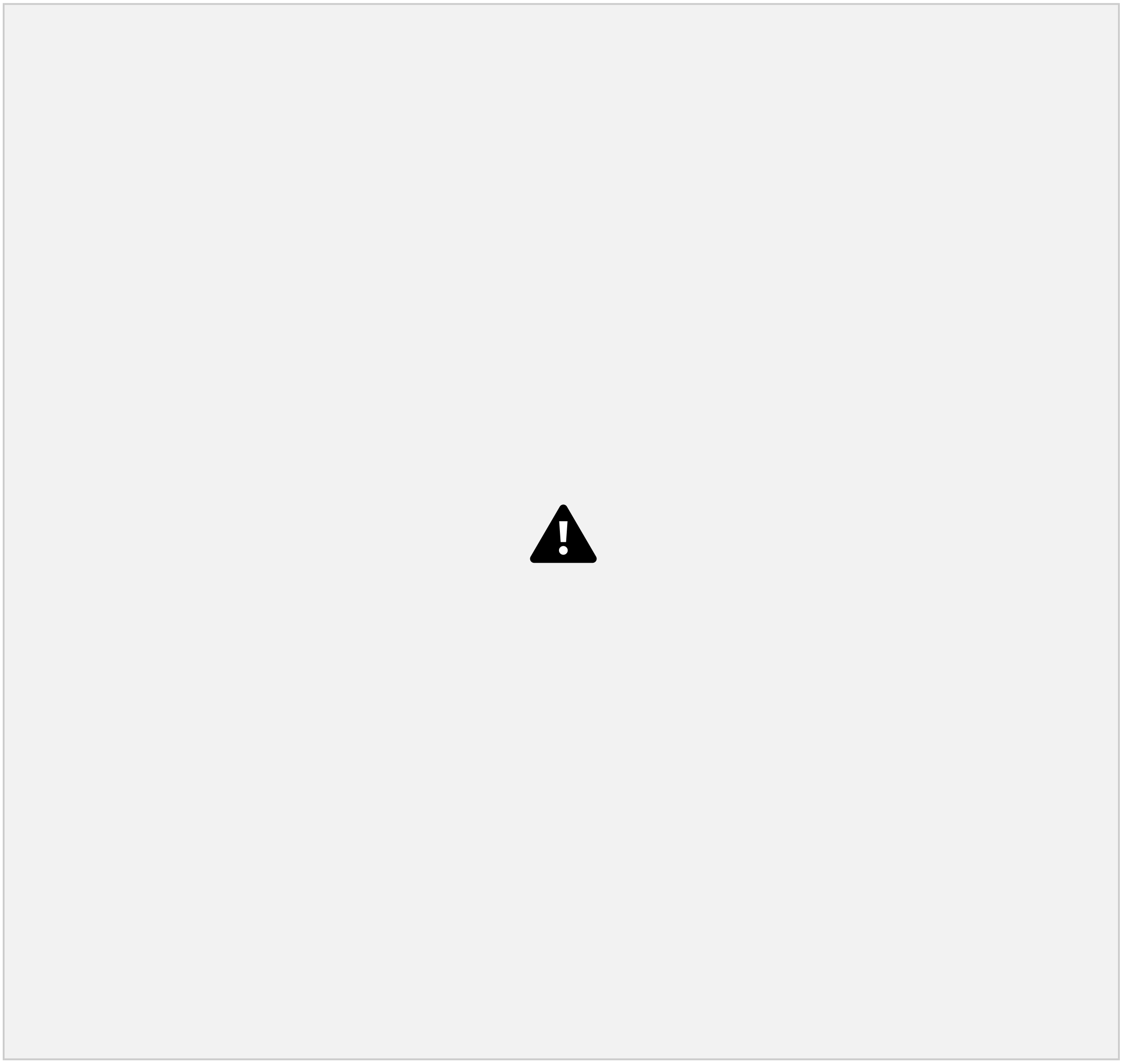


OUTPUT:

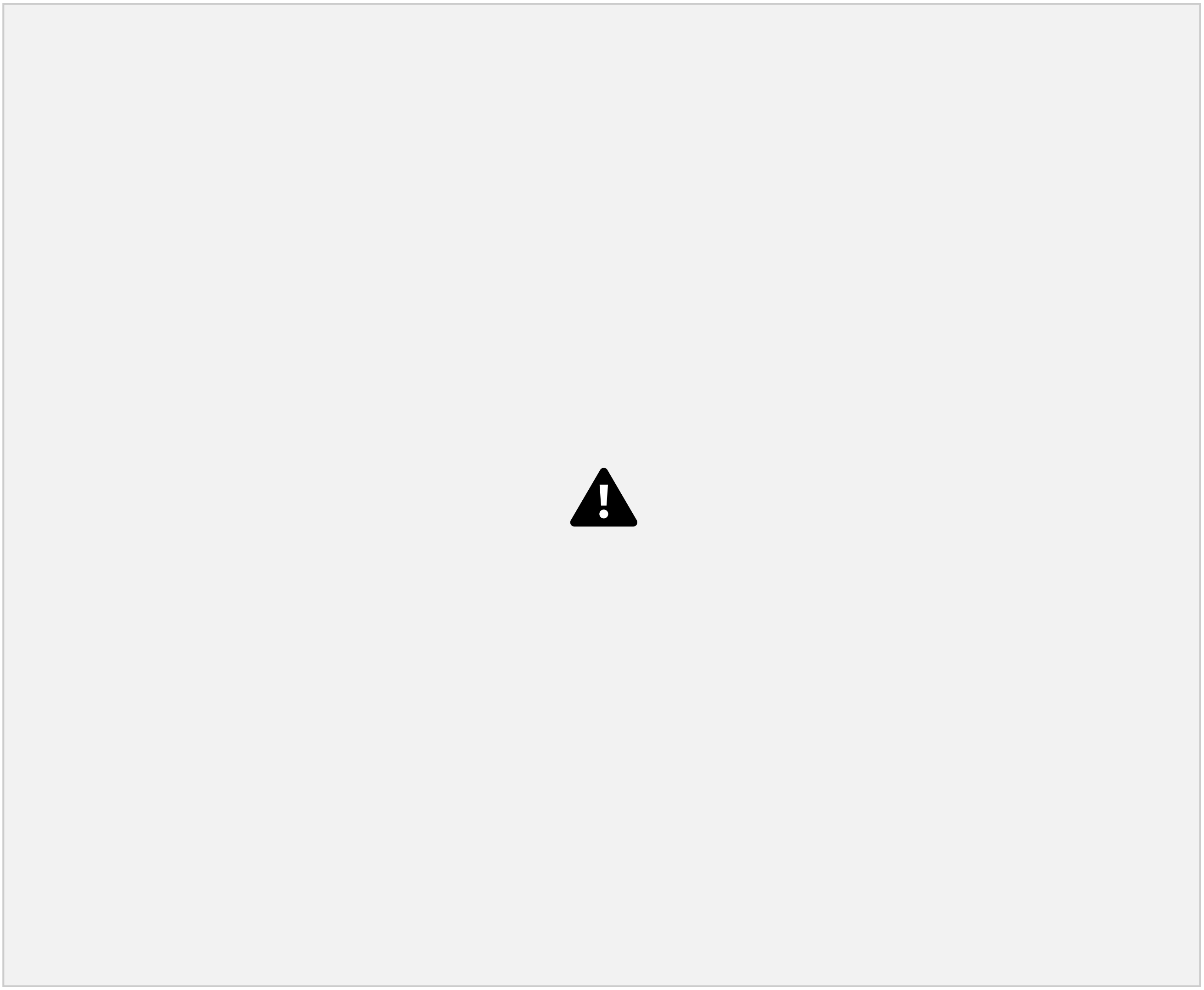
<u>Program 3</u>

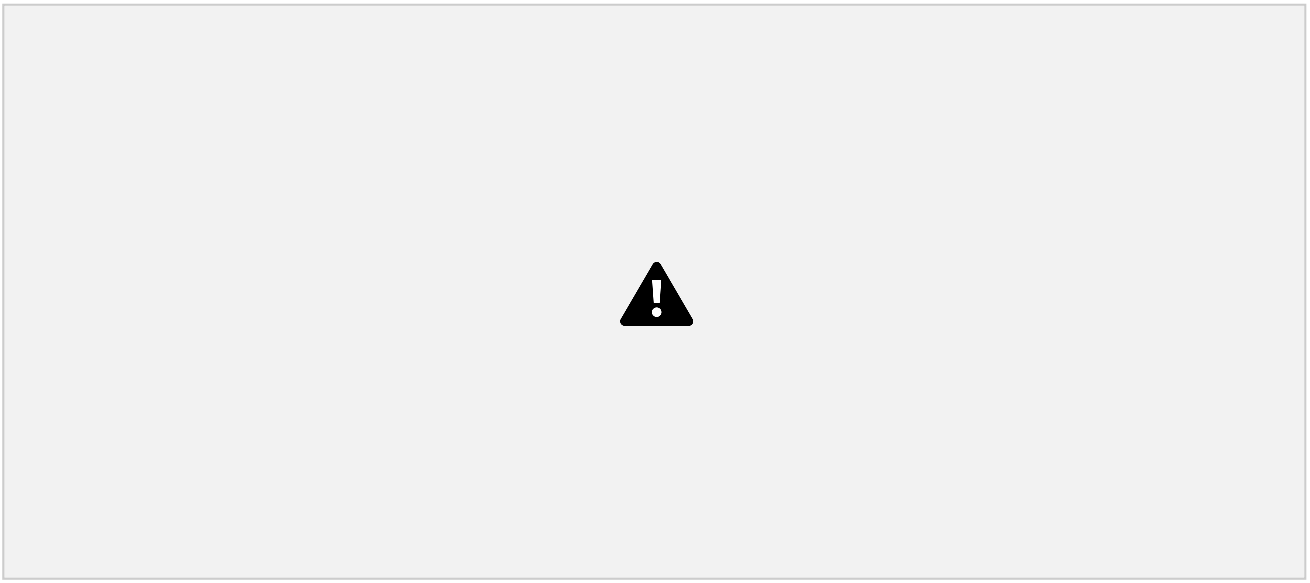
(a) Write a C program to simulate the working of a queue of integers using an array. Provide the following operations: insert, delete,display. The program should print appropriate messages for queue empty and queue overflow conditions.

OUTPUT:



32





Program 3

(b) Write a program in c to simulate the working of a circular queue of integers using an array.

Provide the following operations: insert, delete & display. The program should print appropriate messages for queue empty and queue overflow conditions**.**

OUTPUT:

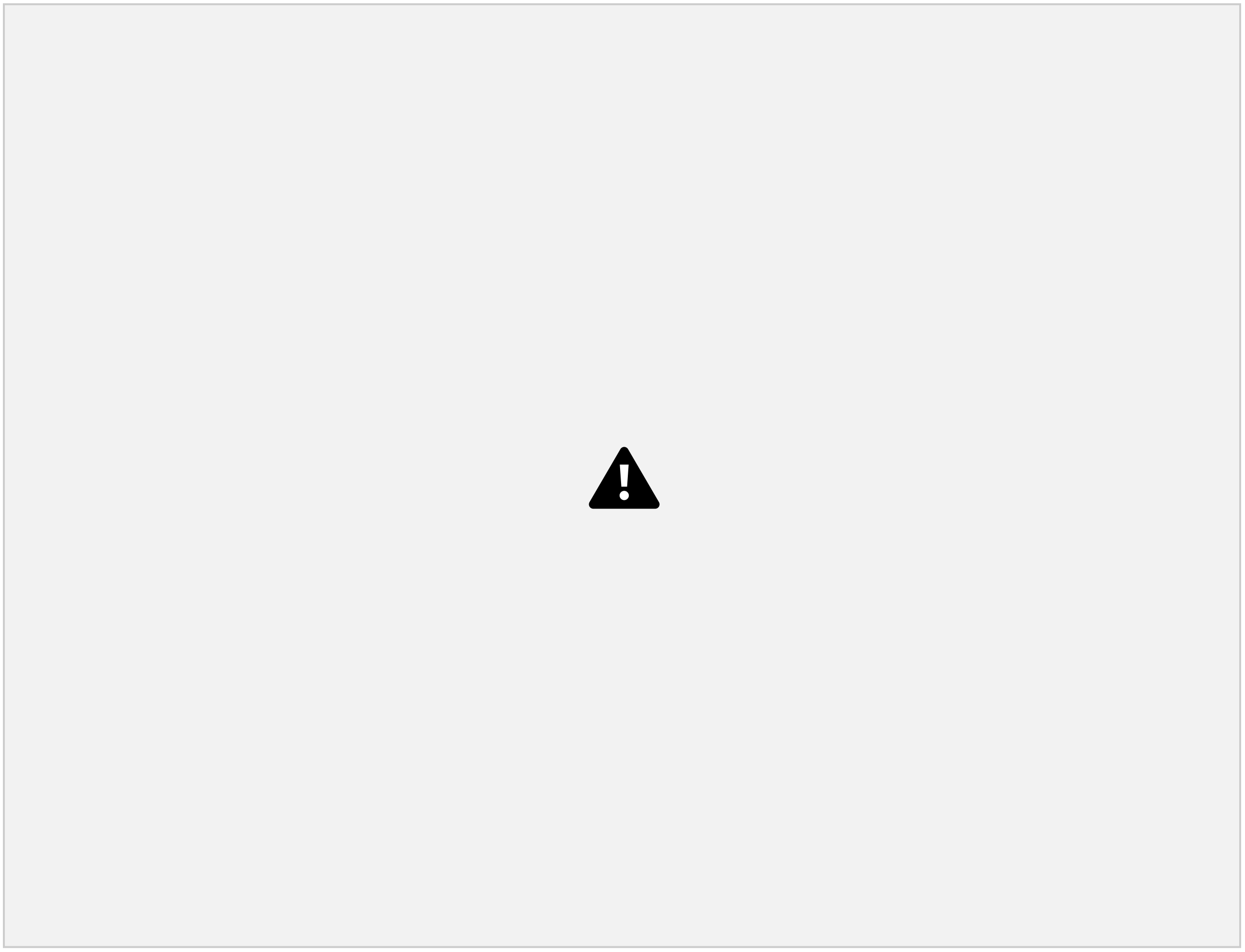
41

Program 4

WAP in c to Implement Singly Linked List with following operations:

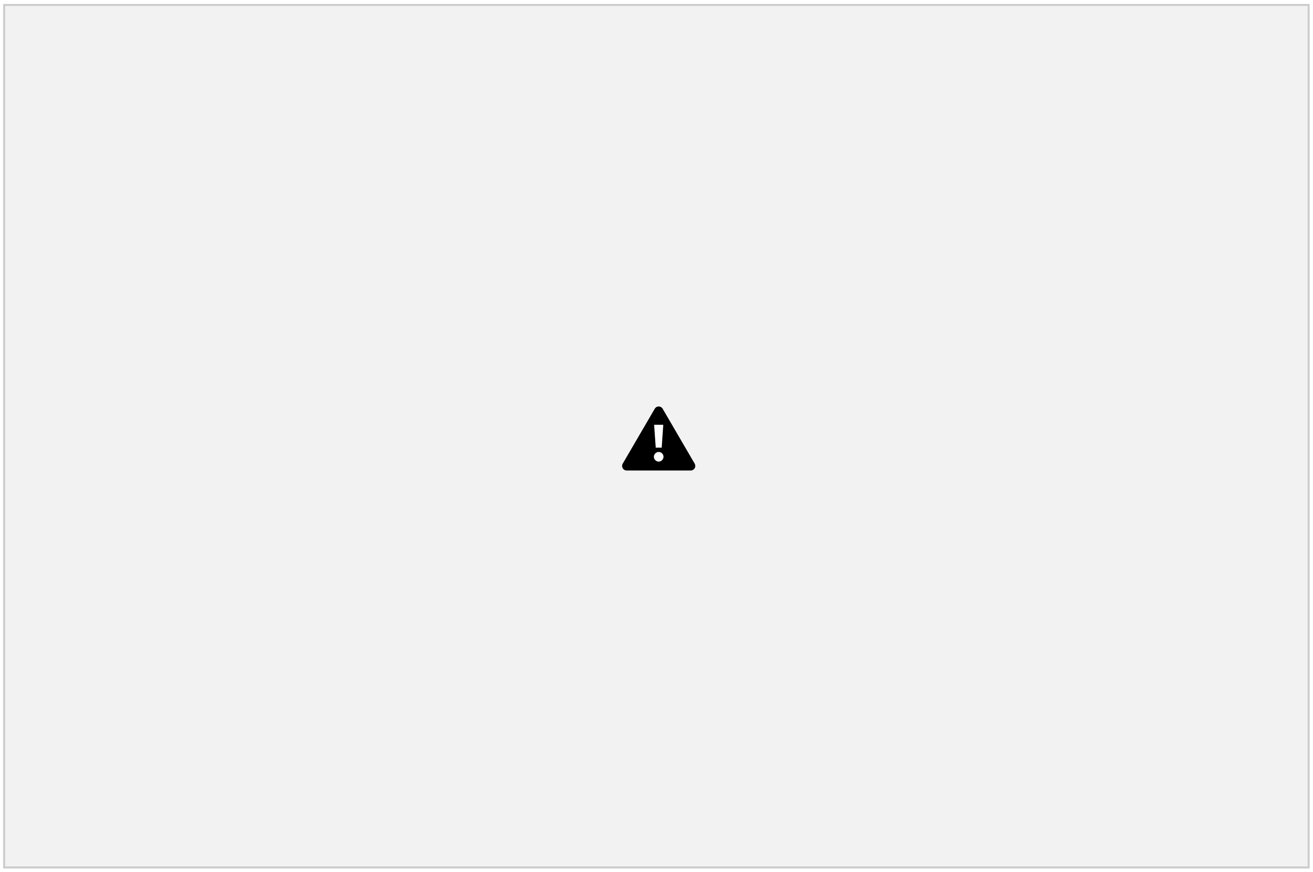
(a) Create a linked list
(b) Insertion of a node at first position and at end of list
(c) Display the contents of the linked list.

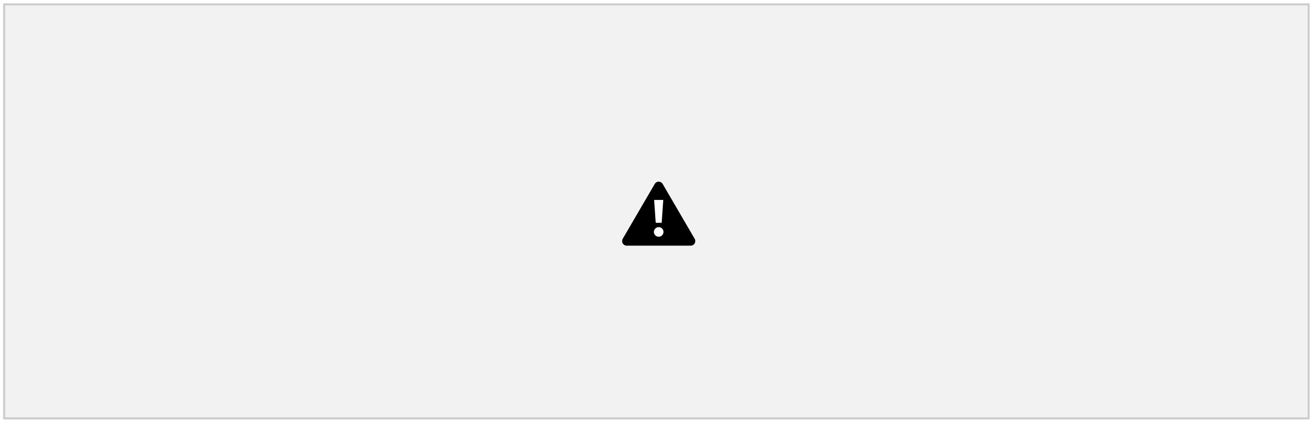
45



46

OUTPUT:



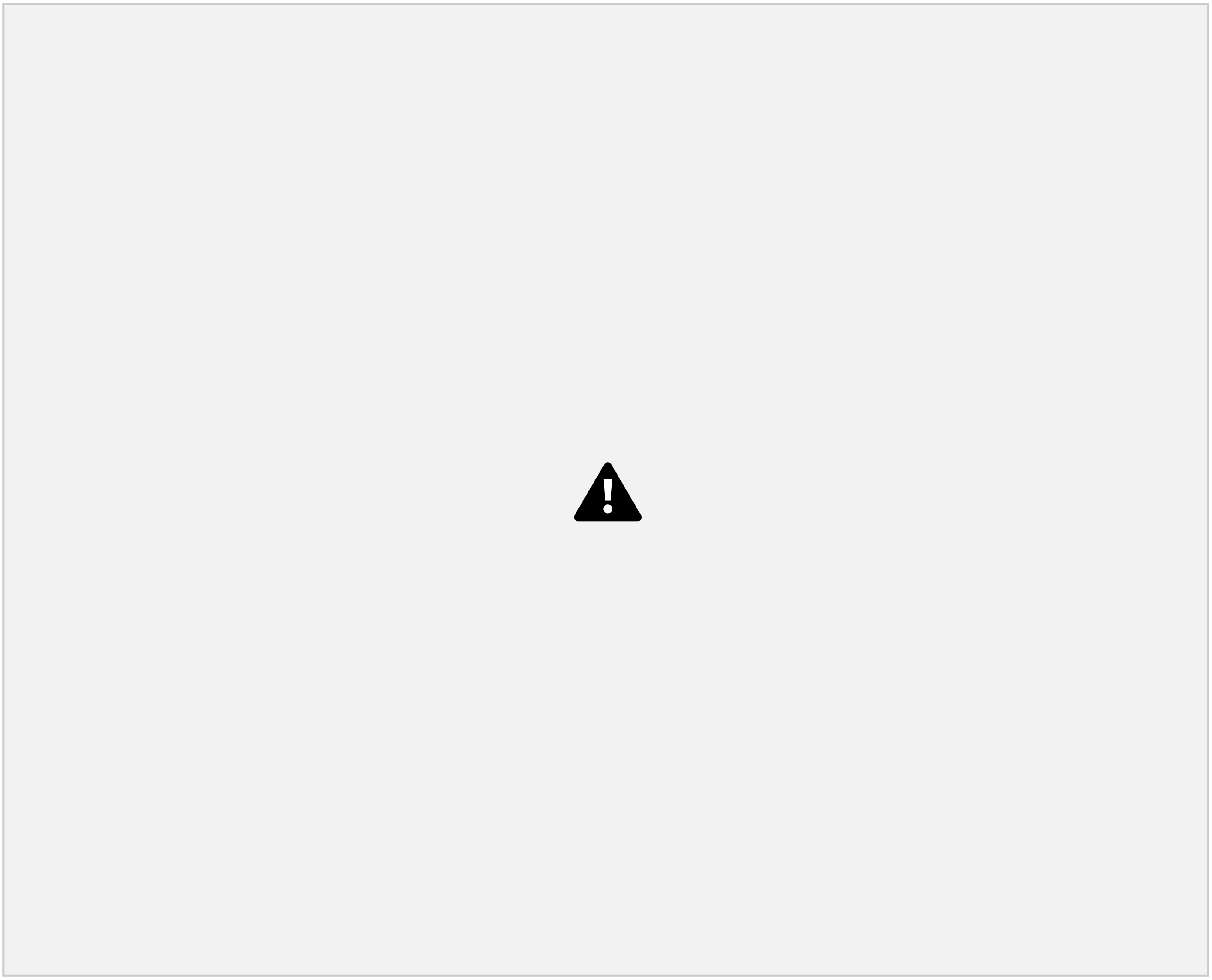


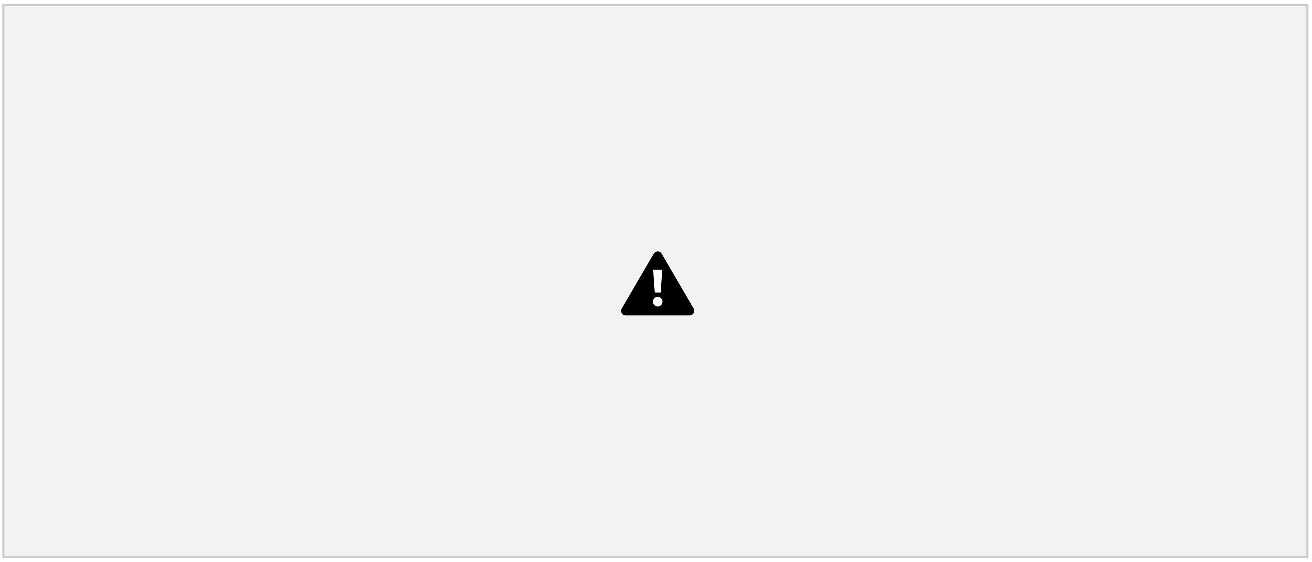
Program 5

WAP in C to Implement Singly Linked List with following operations:

(a) Create a linked list.
(b) Deletion of the first element, specified element and last element in the list.
(c) Display the contents of the linked list.

OUTPUT:





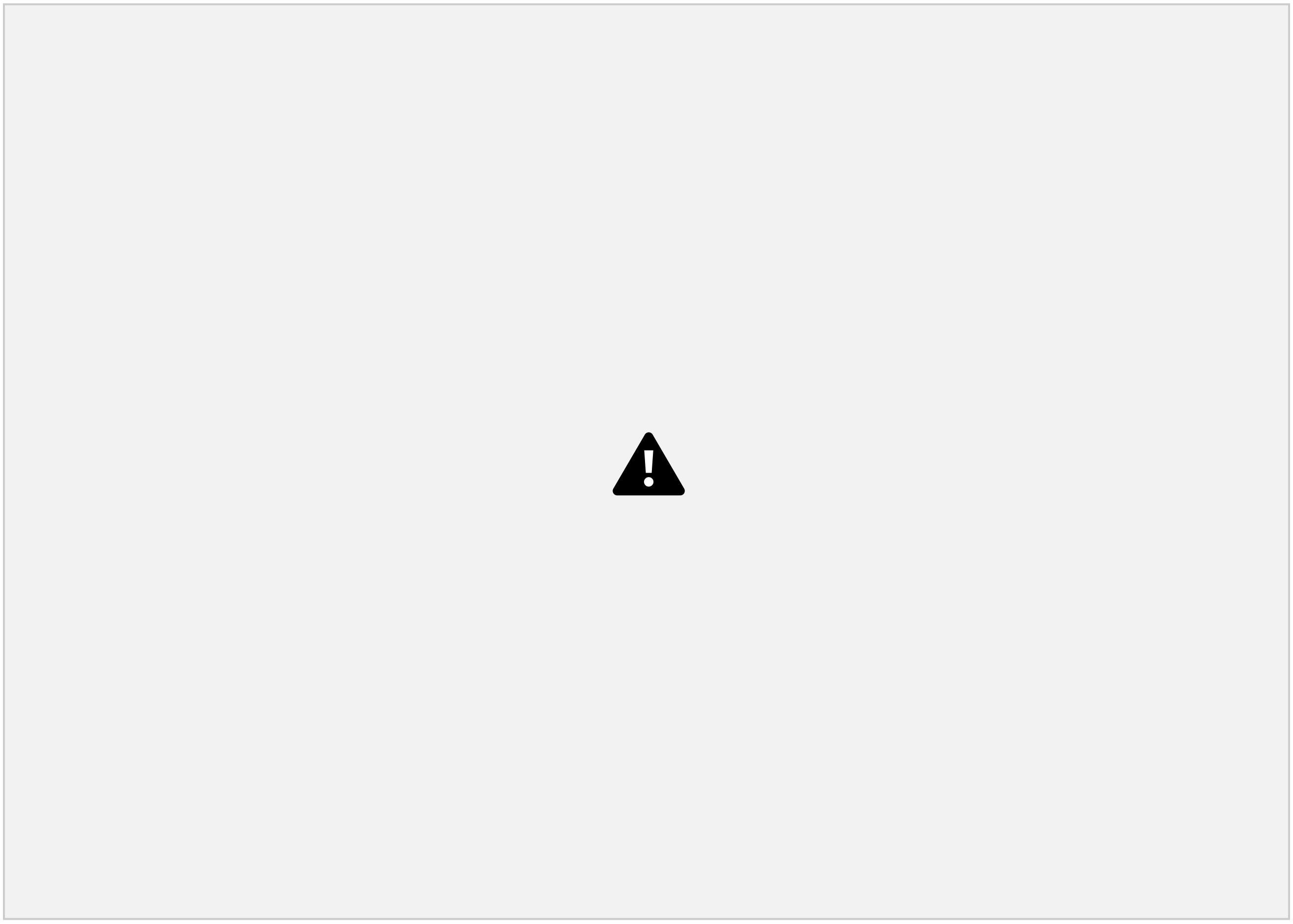
OUTPUT:

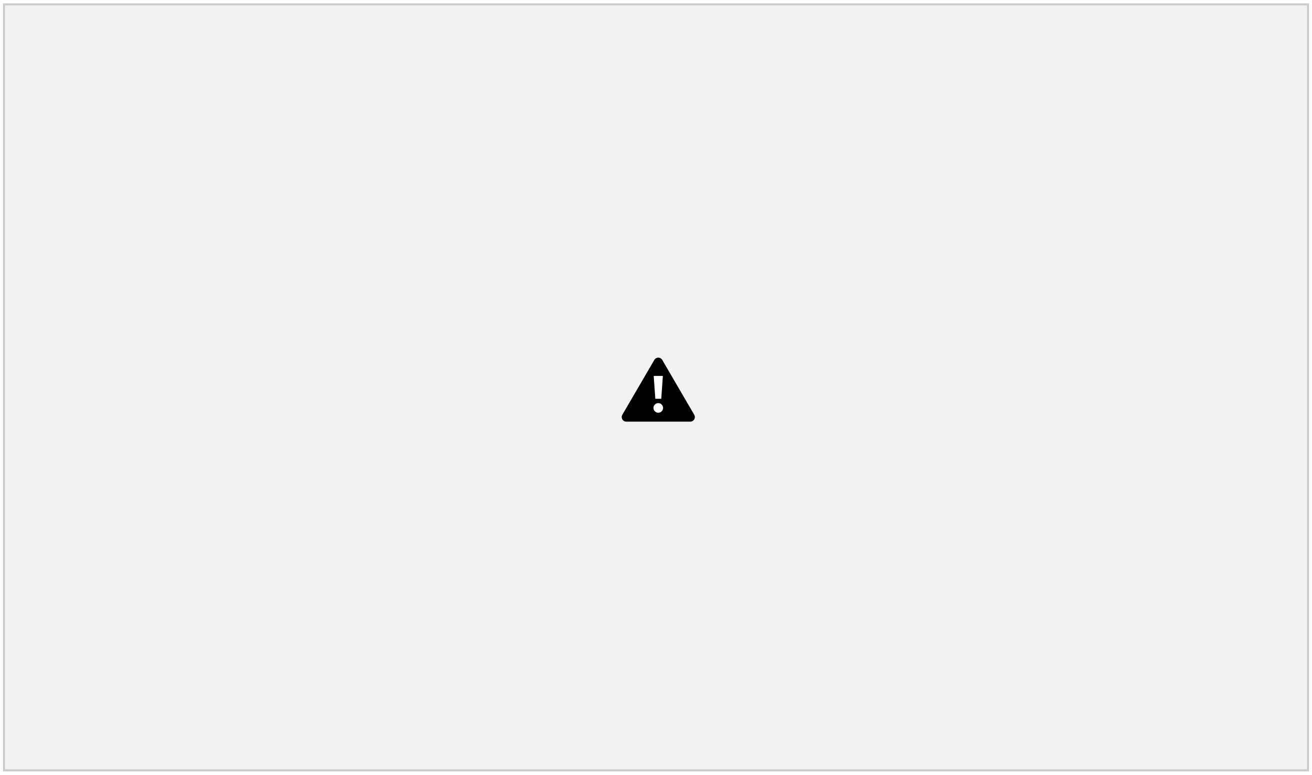
60

Program 6

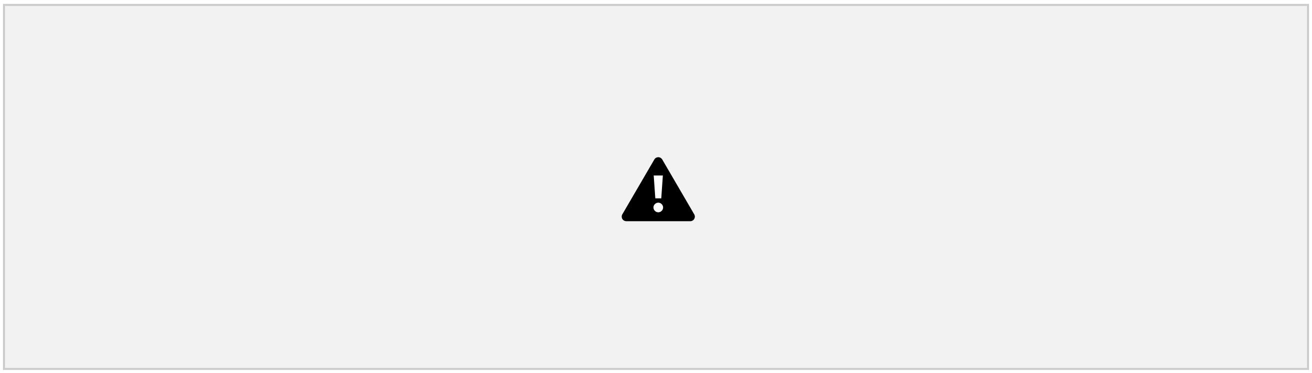
(a) WAP to Implement Single Link List with following operations:

- -Sort the linked list,
- -Reverse the linked list,
- -Concatenation of two linked lists.



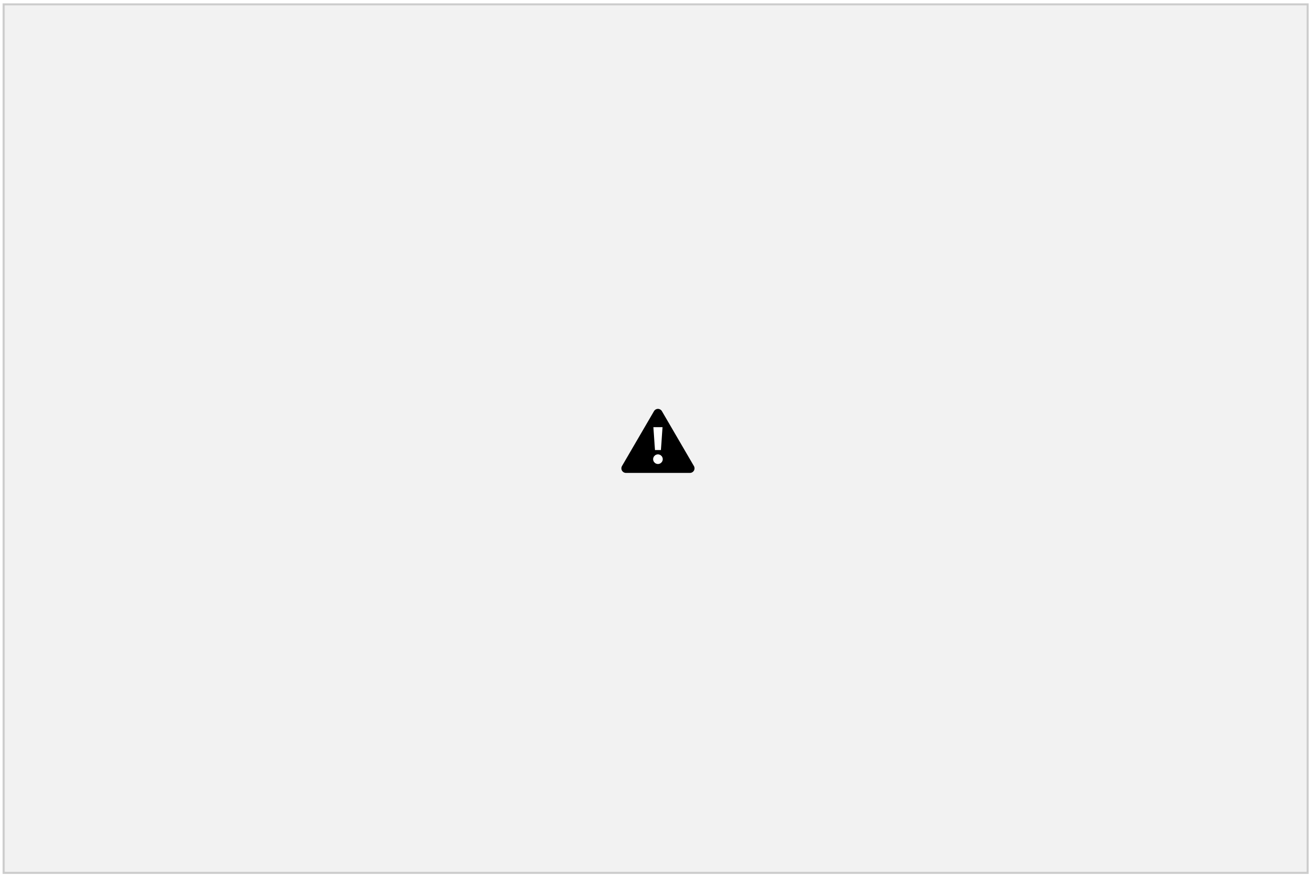


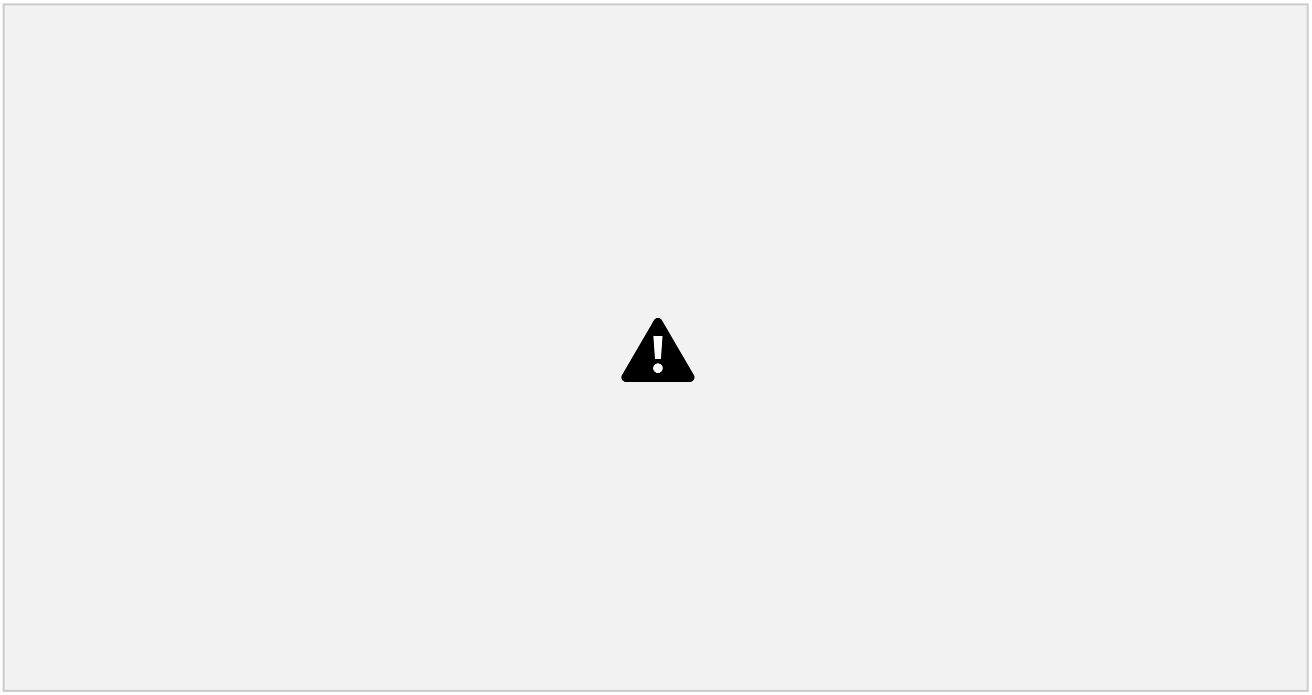
OUTPUT:



(b) WAP to Implement Single Link List to stimulate Stack and Queue operations.

OUTPUT:





Program 7

WAP to implement doubly linked list with operations:

(a) Create a doubly linked list

(b) Insert a new node to the left of the node

(c) Delete a node based on a specific value
(d) Display the contents of a list