Program :

```
#include <iostream>
#include <cstring>
#include <string>
#include <stack>
using namespace std;

struct Node
{
    int data;
    struct Node *next;
} *top = NULL;

void push(int);
void pop();
void display();
void postfix();

void push(int value)
{
    struct Node *newNode;
    newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    if (top == NULL)
        newNode->next = NULL;
    else
        newNode->next = top;
    top = newNode;
    cout << ("\nInsertion is Success!!!\n");
}

void pop()
{
    if (top == NULL)
        cout << ("\nStack is Empty!!!");
    else
    {
        struct Node *temp = top;
        cout << ("\nDeleted element: %d", temp->data);
        top = temp->next;
        free(temp);
    }
}

void display()
{
    if (top == NULL)
        cout << ("\nStack is Empty!!!\n");
    else
    {
        struct Node *temp = top;
        while (temp->next != NULL)
        {
            cout << ("\t", temp->data) << ", ";
```

```
            temp = temp->next;
        }
        cout << ("\t", temp->data) << ", ";
    }
}

struct Stack
{
    int top;
    unsigned capacity;
    int *array;
};
// Stack Operations
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()
{
    int choice, value;
    cout << ("\n********* MENU **********\n");
    cout << "* 1. Push in stack      *" << endl;
    cout << "* 2. Pop from stack     *" << endl;
    cout << "* 3. Display stack      *" << endl;
    cout << "* 4. Postfix Evaluation *" << endl;
    cout << "* 5. Exit               *" << endl;
    cout << "************************" << endl;

    do
    {
        cout << ("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            cout << ("Enter the value to be insert: \n");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            pop();
            break;
        case 3:
            display();
            break;
        case 4:
        {
            char exp[] = "10 20 * 30 60 10 / - +";
            cout << "Postfix Evaluation: " << evaluatePostfix(exp)<<"\n";
            return 0;
```

```
        }
        case 5:
            exit(0);
            break;
        default:
            cout << ("\nWrong selection!!! Please try again!!!\n");
        }
    } while (choice != 4);
    return 0;
}
```

Output :