DSA LAB

Week 4

Udeet Mittal

CSE C3

Roll Number 64

I. SOLVED EXERCISE:

1) Program for evaluation of postfix expression in C

filename: eval_postfix_fun.h

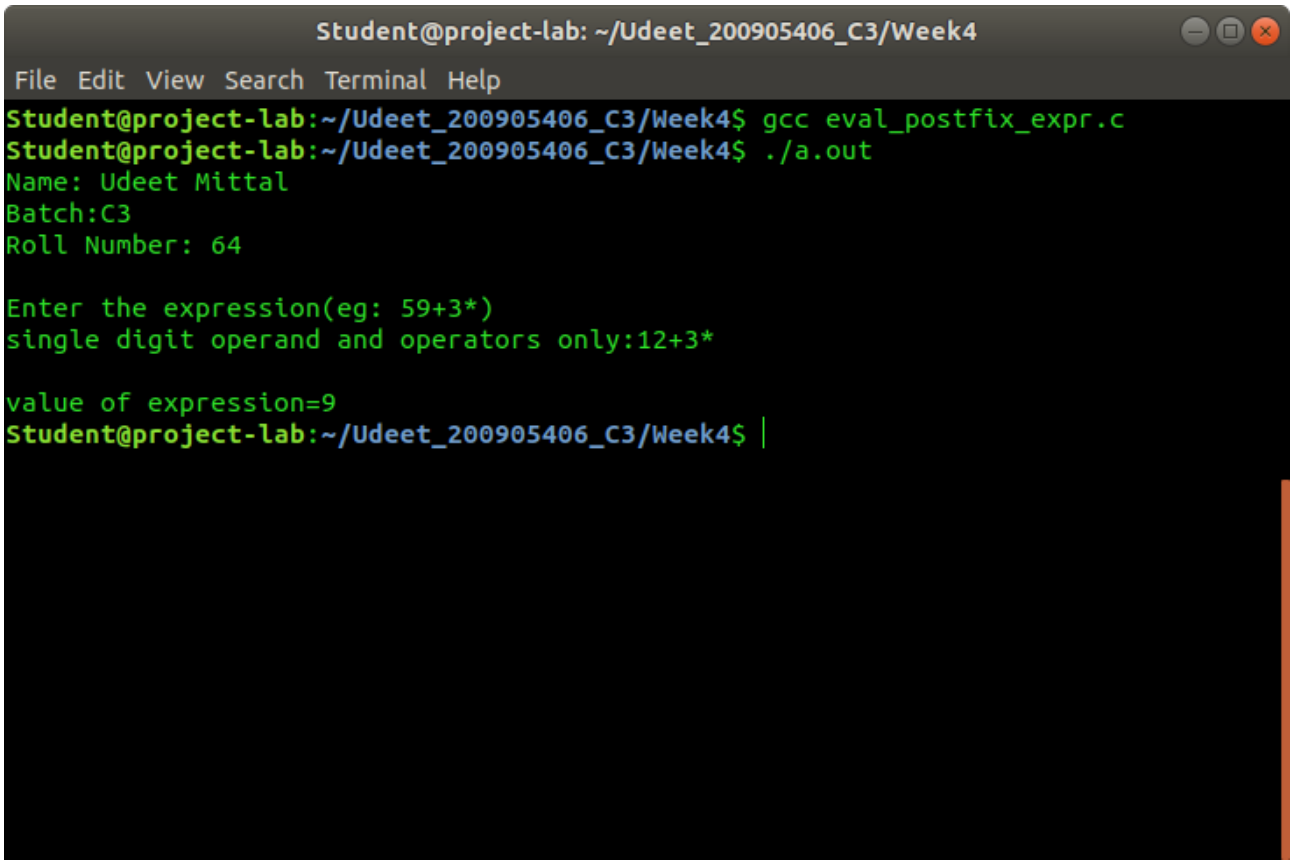
```
#define MAX 20

typedef struct stack
{
 int data[MAX];
 int top;
}stack;

void init(stack *);
int empty(stack *);
int full(stack *);
int pop(stack *);
void push(stack *,int);
int evaluate(char x,int op1,int op2);

int evaluate(char x,int op1,int op2)
{
 if(x=='+')
 return(op1+op2);
 if(x=='-')
 return(op1-op2);
 if(x=='*')
 return(op1*op2);
 if(x=='/')
 return(op1/op2);
 if(x=='%')
 return(op1%op2);
}

void init(stack *s)
{
```

```
 s->top=-1;
}

int empty(stack *s)
{
 if(s->top==-1)
 return(1);
 return(0);
}

int full(stack *s)
{
 if(s->top==MAX-1)
 return(1);
 return(0);
}

void push(stack *s,int x)
{
 s->top=s->top+1;
 s->data[s->top]=x;
}

int pop(stack *s)
{
 int x;
 x=s->data[s->top];

 s->top=s->top-1;
 return(x);
}
```

filename: eval_postfix_expr.c

```
#include<stdio.h>
#include "eval_postfix_fun.h"
#include<ctype.h>
int main()
{
 stack s;
 char x;
 int op1,op2,val;
 init(&s);
 printf("Name: Udeet Mittal\nBatch:C3\nRoll Number: 64\n\n");
 printf("Enter the expression(eg: 59+3*)\nsingle digit operand and operators only:");
 while((x=getchar())!='\n')
 {
 if(isdigit(x))
    push(&s,x-'0');
```

```
 else
 {
 op2=pop(&s);
 op1=pop(&s);
 val=evaluate(x,op1,op2);
 push(&s,val);
 }
 }
 val=pop(&s);
 printf("\nvalue of expression=%d",val);
 printf("\n");
 return 0;
}
```



Questions for Lab4

Write a C program to:

1) Evaluate a given prefix expression using stack.

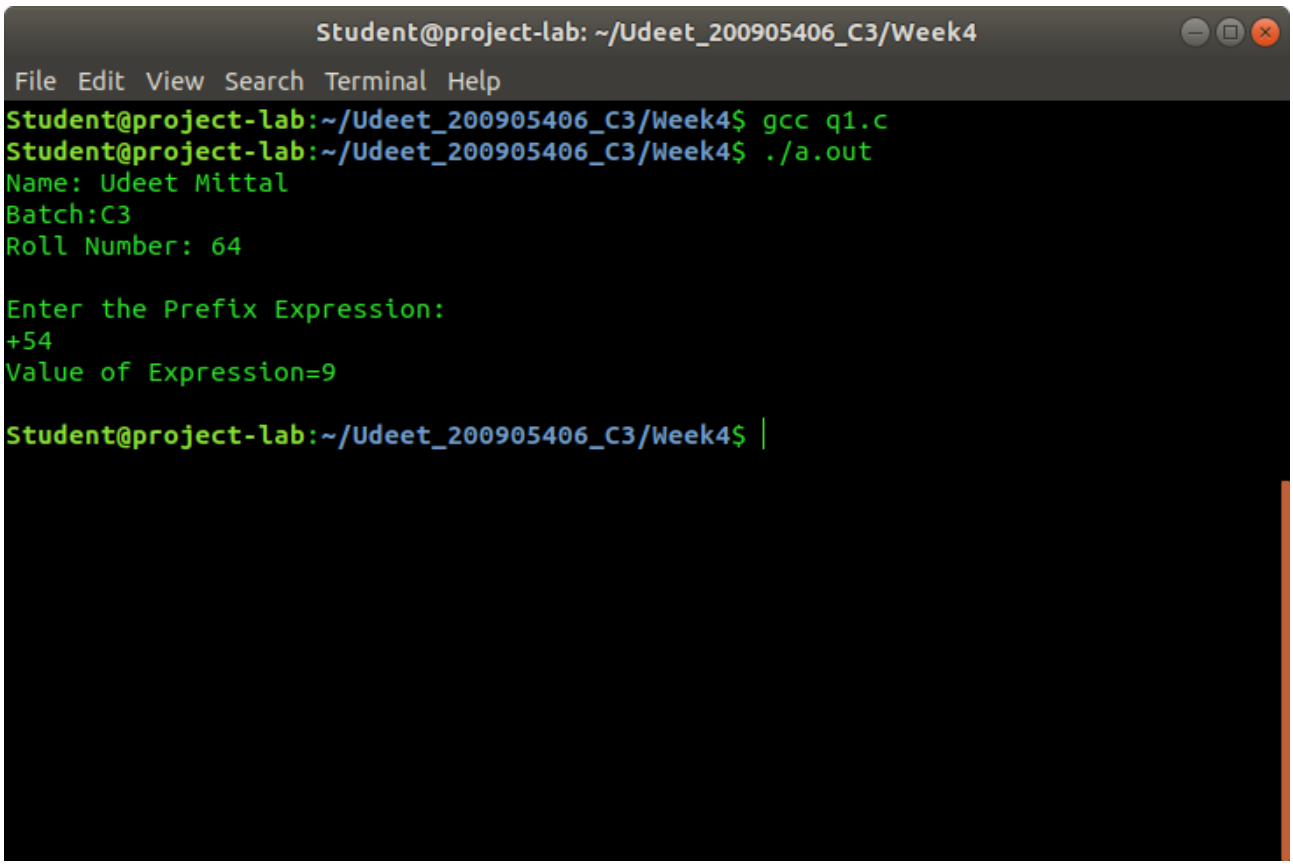
Filename: prefix.h

```
#define MAX 20

typedef struct
{
int data[MAX];
int top;
}stack;

void init(stack *);
int empty(stack *);
int full(stack *);
int pop(stack *);
void push(stack *, int);
int evaluate(char x, int op1, int op2);

int evaluate(char x, int op1, int op2)
{
if (x == '+')
return (op1 + op2);
if (x == '-')
return (op1 - op2);
if (x == '*')
return (op1 * op2);
if (x == '/')
return (op1 / op2);
if (x == '%')
return (op1 % op2);
}

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

int empty(stack *s)
{
if (s->top == -1)
return (1);
return (0);
}

int full(stack *s)
{
if (s->top == MAX - 1)
return (1);
return (0);
}

void push(stack *s, int x)
{
s->top = s->top + 1;
s->data[s->top] = x;
}

int pop(stack *s)
```

```
{
int x;
x = s->data[s->top];
s->top = s->top - 1;
return (x);
}
```

filename: q1.c

```
#include <stdio.h>
#include <string.h>
#include "prefix.h"
#include <ctype.h>

int main()
{
 printf("Name: Udeet Mittal\nBatch:C3\nRoll Number: 64\n\n");
stack st;
stack *s = &st;
init(s);
char x;
int op1, op2, val;
printf("Enter the Prefix Expression:\n");
char expr[MAX];
scanf("%s", expr);
for (int i = (strlen(expr)-1); i >= 0; i--)
{
x = expr[i];
if(isdigit(x))
{
push(s, x-'0');
}
else
{
op1 = pop(s);
op2 = pop(s);
val = evaluate(x,op1,op2);
push(s, val);
}
}
val = pop(s);
printf("Value of Expression=%d \n", val);
printf("\n");
return 0;
}
```



2.Convert an infix expression to prefix.

Filename: infix.h

```
#define MAX 20

typedef struct
{
int item[MAX];
int top;
}stack;

void initialize(stack *);
int pop(stack *);
void push(stack *, int);
int isOperator(char symbol);
int isp(char symbol);
int icp(char symbol);

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

void push(stack *s, int x)
```

```
{
s->top = s->top + 1;
s->item[s->top] = x;
}

int pop(stack *s)
{
int x;
x = s->item[s->top];
s->top = s->top - 1;
return (x);
}

int icp(char symbol)
{
switch (symbol)
{
case '+':
case '-':
return 12;
break;
case '*':
case '/':
case '%':
return 13;
break;
case '(':
return 19;
break;
case ')':
return 20;
break;
}
}

int isp(char symbol)
{
switch (symbol)
{
case '+':
case '-':
return 12;
break;
case '*':
case '/':
case '%':
return 13;
break;
case '(':
return 19;
break;
case ')':
```

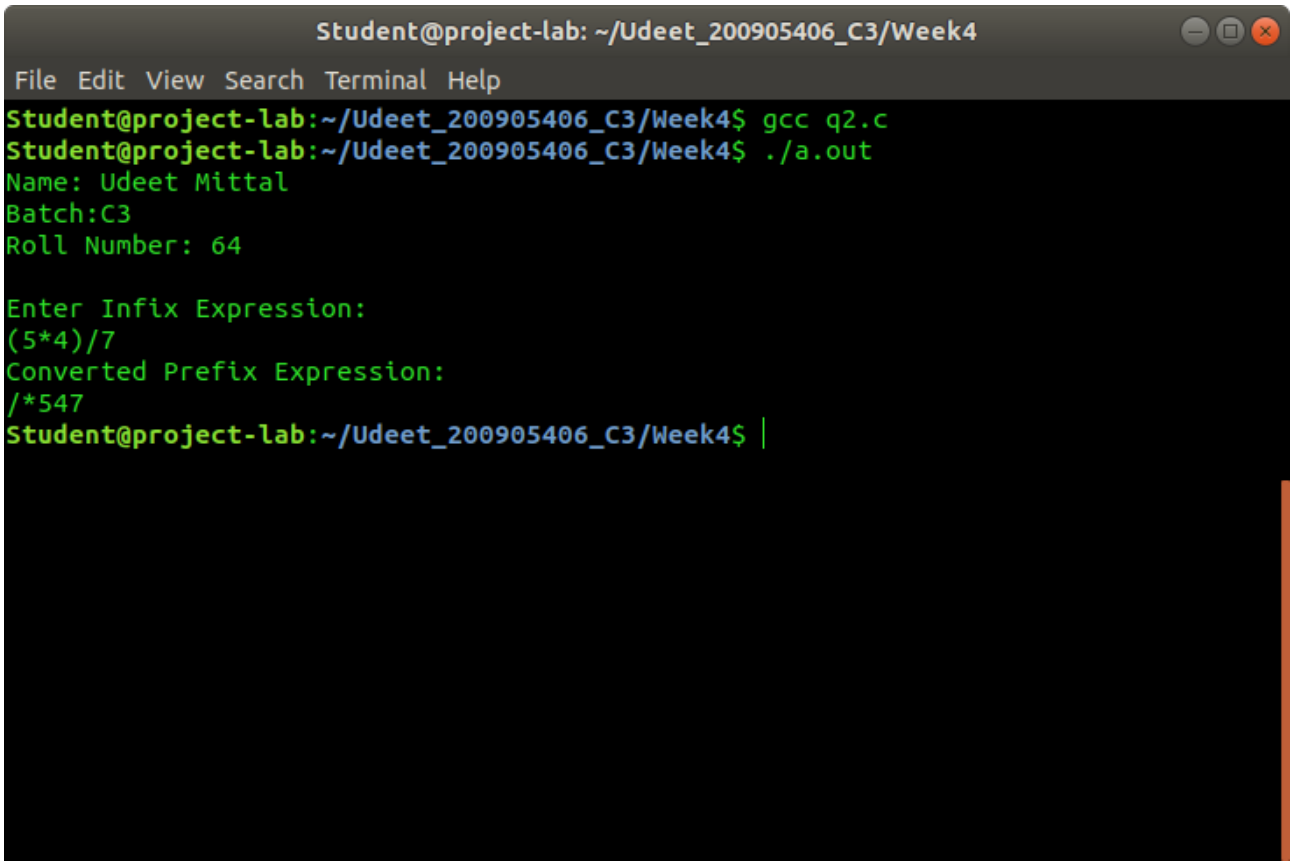
```
return 0;
break;
}
}

int isOperator(char symbol)
{
switch (symbol)
{
case '+':
case '-':
case '*':
case '/':
case '%':
case '(':
case ')':
return 1;
break;
default:
return 0;
}
}
```

filename: q2.c

```
#include <stdio.h>
#include <string.h>
#include "infix.h"
void infixtoprefix(char infix[20], char prefix[20], stack *s);
char* reverse(char str[]);

int main()
{
 printf("Name: Udeet Mittal\nBatch:C3\nRoll Number: 64\n\n");
 stack st;
stack *s = &st;
initialize(s);
char infix[20], prefix[20], temp;
printf("Enter Infix Expression:\n");
scanf("%s", infix);
infixtoprefix(infix, prefix, s);
strcpy(prefix, reverse(prefix));
printf("Converted Prefix Expression:\n%s \n", prefix);
return 0;
}

void infixtoprefix(char infix[20], char prefix[20], stack *s)
{
int j = 0;
char symbol;
```

```
push(s, '\0');
strcpy(infix, reverse(infix));
for (int i = 0; i < strlen(infix); i++)
{
symbol = infix[i];
if (isOperator(symbol) == 0)
  {
prefix[j] = symbol;
j++;
  }
else
{
if (symbol == ')')
  {
push(s, symbol);
   }
else if (symbol == '(')
{
while (s->item[s->top] != ')')
  {
 prefix[j] = pop(s);
 j++;
   }
pop(s);
}
else
{
if (isp(s->item[s->top]) <= icp(symbol))
  {
push(s, symbol);
  }
else
  {
while (isp(s->item[s->top]) > icp(symbol))
        {
prefix[j] = pop(s);
j++;
        }
push(s, symbol);
  }
 }
}

}
while (s->item[s->top] != '\0')
   {
prefix[j] = pop(s);
j++;
  }
prefix[j] = '\0';
}
```

```
char* reverse(char str[]){
int length = strlen(str);
char temp;
for(int i=0, j=length-1; i<length/2; i++,j--){
temp = str[i];
str[i]=str[j];
str[j]=temp;
}
return str;
}
```



3.Implement two stacks in an array.

filename: stack.h

```
#include <stdio.h>
#include <stdlib.h>
#define MAX (100)
#define TRUE (1)
#define FALSE (0)
#define SUCCESS (1)
#define FAILED (0)
```

```
typedef struct {
 char item[MAX];
 int top;
}stack;
int isEmpty(stack*);
int isFull(stack*);
int push(stack*, char);
char pop(stack*);
void display(stack*);
stack* new_stack();

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

int isFull(stack *s)
{
 if(s->top == MAX - 1)
 return TRUE;
 return FALSE;
}

int push(stack *s, char elem)
{
 if(isFull(s))
 return FAILED;
 s->item[++s->top] = elem;
 return SUCCESS;
}

char pop(stack *s)
{
 if(isEmpty(s))
 return FAILED;
 return(s->item[s->top--]);
}

void display(stack *s)
{
 if(isEmpty(s)) return;
 int i;
 for(i = 0; i <= s->top; i++)
 printf("%c ", s->item[i]);
 printf("\n");
}

stack* new_stack()
{
    stack str;
```

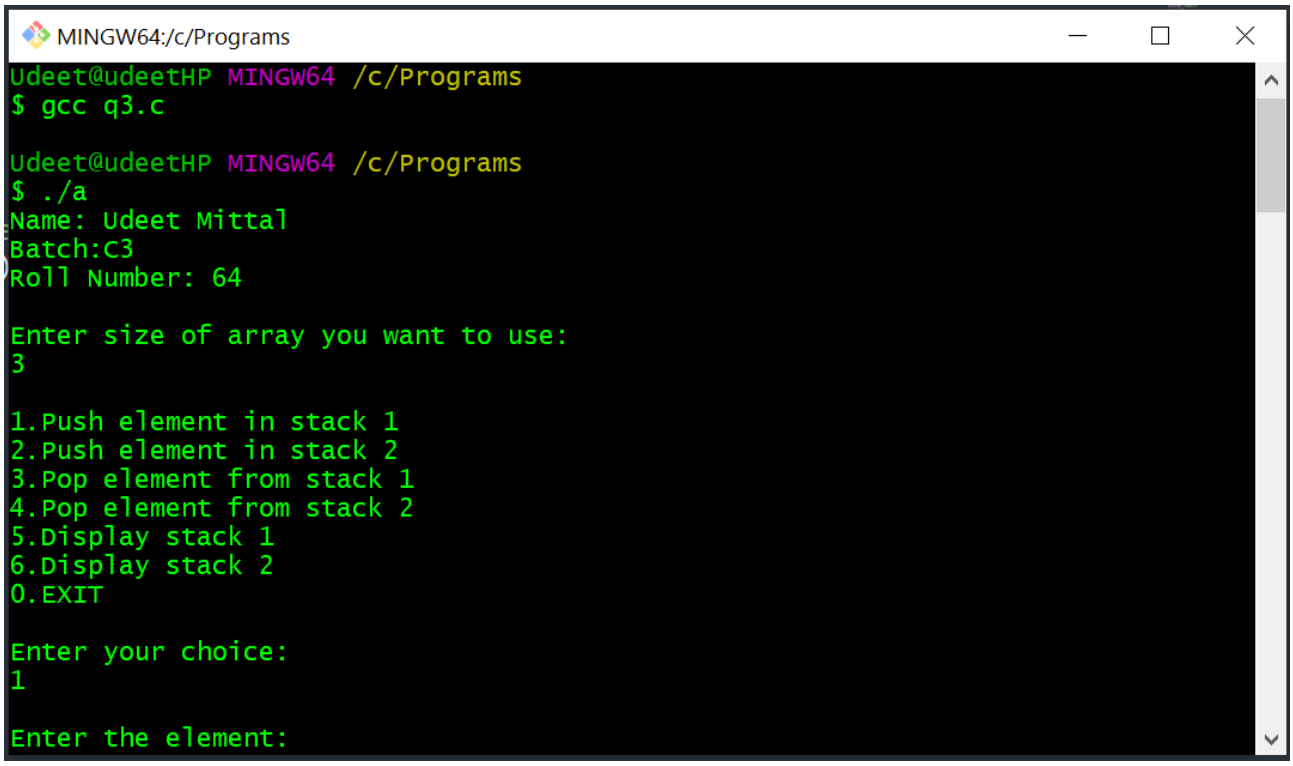
```
 stack *s=&str;
 s->top = -1;
 return s;
}
```

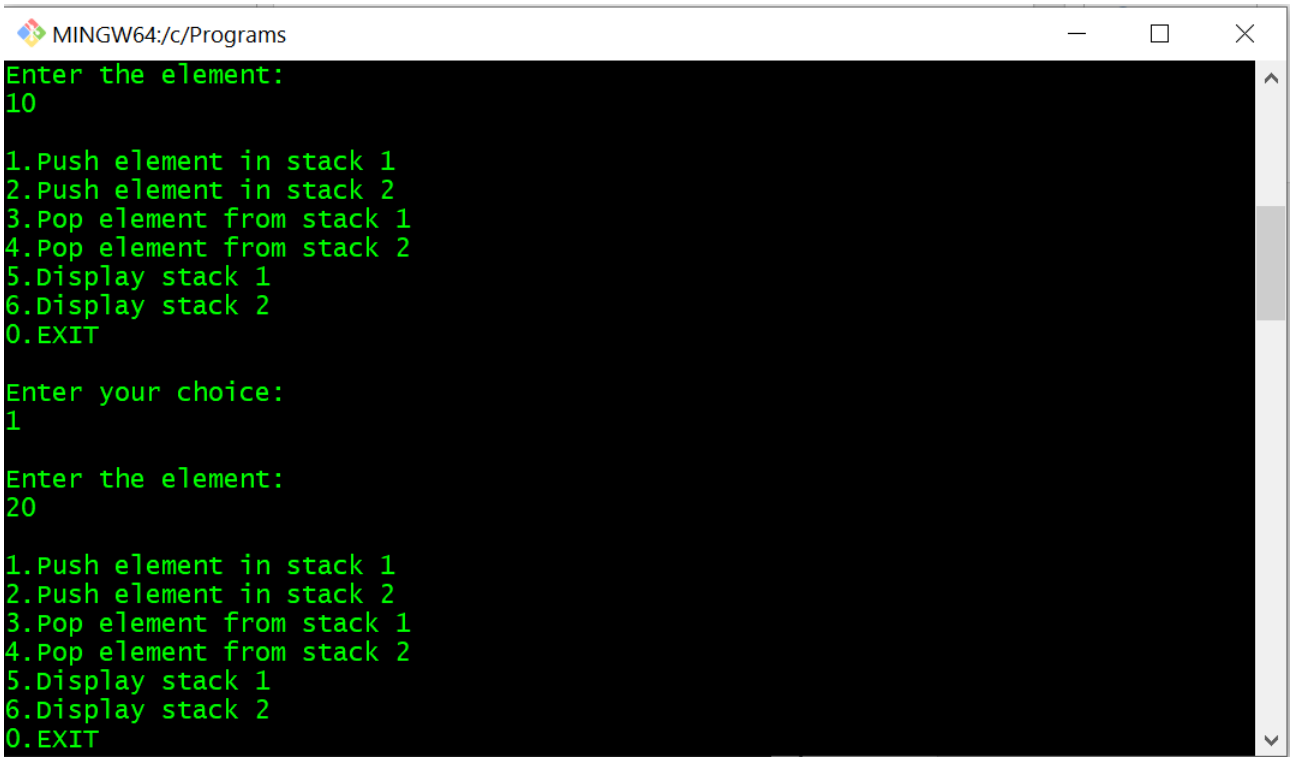
Filename: q3.c

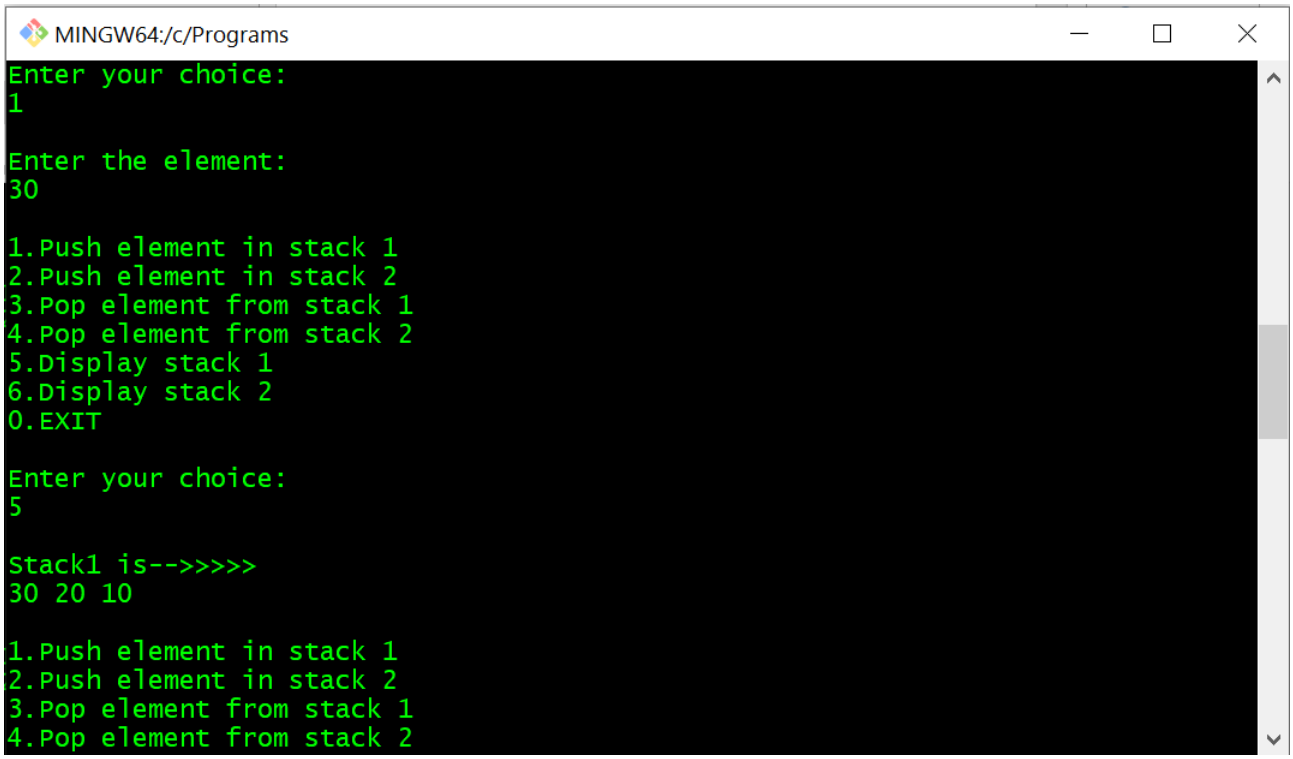
```
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"

int main()
{
 int n,top1,top2,ch=1,a,i,arr[100];
printf("Name: Udeet Mittal\nBatch:C3\nRoll Number: 64\n\n");
 printf("Enter size of array you want to use\n");
 scanf("%d",&n);top1=-1;
 top2=n;
 while(ch!=0)
 {
 printf("\n1.Push element in stack 1\n");
 printf("2.Push element in stack 2\n");
 printf("3.Pop element from stack 1\n");
 printf("4.Pop element from stack 2\n");
 printf("5.Display stack 1\n");
 printf("6.Display stack 2\n");
 printf("0.EXIT\n");
 printf("\nEnter your choice:\n");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1:
 {
 printf("\nEnter the element:\n");
 scanf("%d",&a);
 if(top1!=(top2-1))
 arr[++top1]=a;
 else
 printf("Overflow\n");break;
 }
 case 2:
 {
 printf("\nEnter the element\n");
 scanf("%d",&a);
 if(top2!=(top1+1))
 arr[--top2]=a;
 else
 printf("Overflow\n");
 break;
 }
```

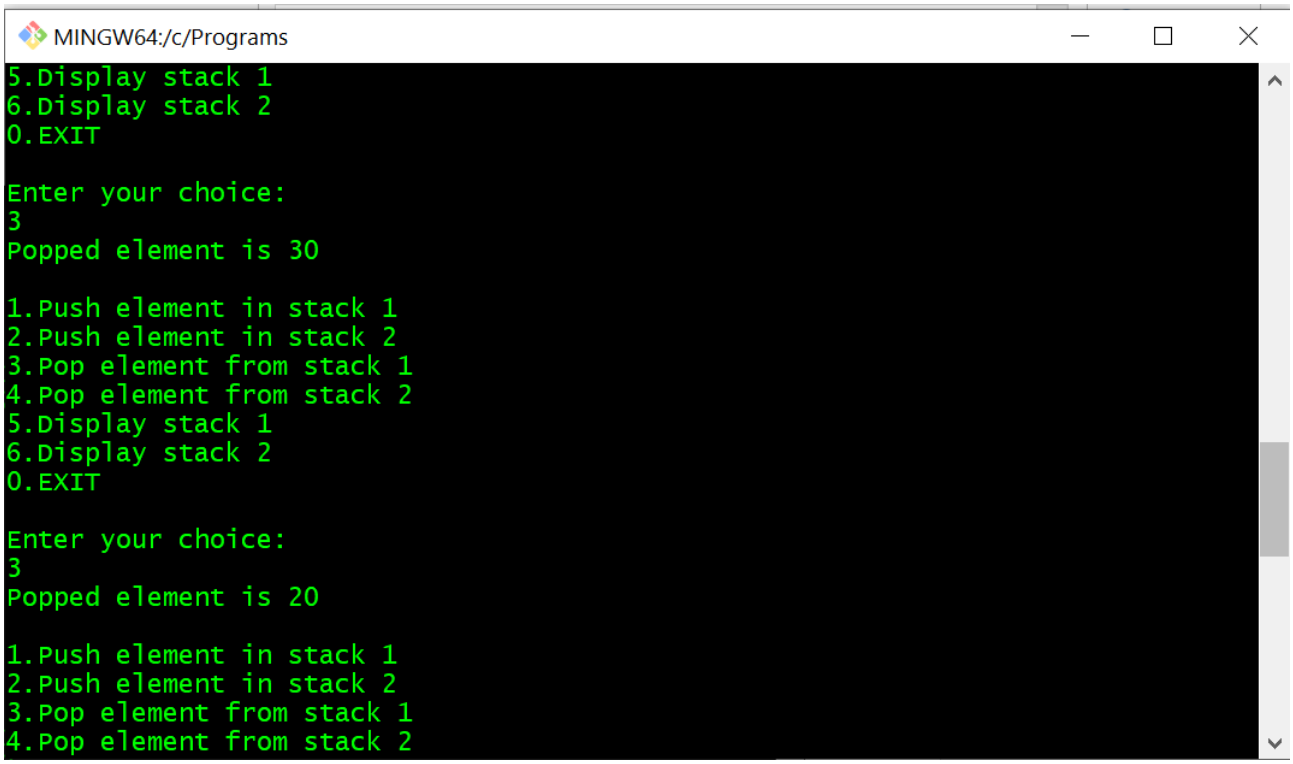
```
case 3:
{
if(top1==-1)
printf("\nStack1 is empty\n");
else
{
a=arr[top1--];
printf("Popped element is %d\n",a);
}
break;
}
case 4:
{
if(top2==n)
printf("\nStack2 is empty\n");
else
{
a=arr[top2++];
printf("Popped element is %d\n",a);
}
break;
}
case 5:
{
if(top1==-1)
printf("\nStack1 is empty\n");
else
{
printf("\nStack1 is-->>>>>\n");
for(i=top1;i>=0;i--)
printf("%d ",arr[i]);
printf("\n");
}
break;
}
case 6:
{
if(top2==n)
printf("\nStack2 is empty\n");
else
{
printf("\nStack2 is-->>>>>>\n");
for(i=top2;i<=n-1;i++)
printf("%d ",arr[i]);
printf("\n");
}
break;
}
case 0:
break;
}
}return 0;
```

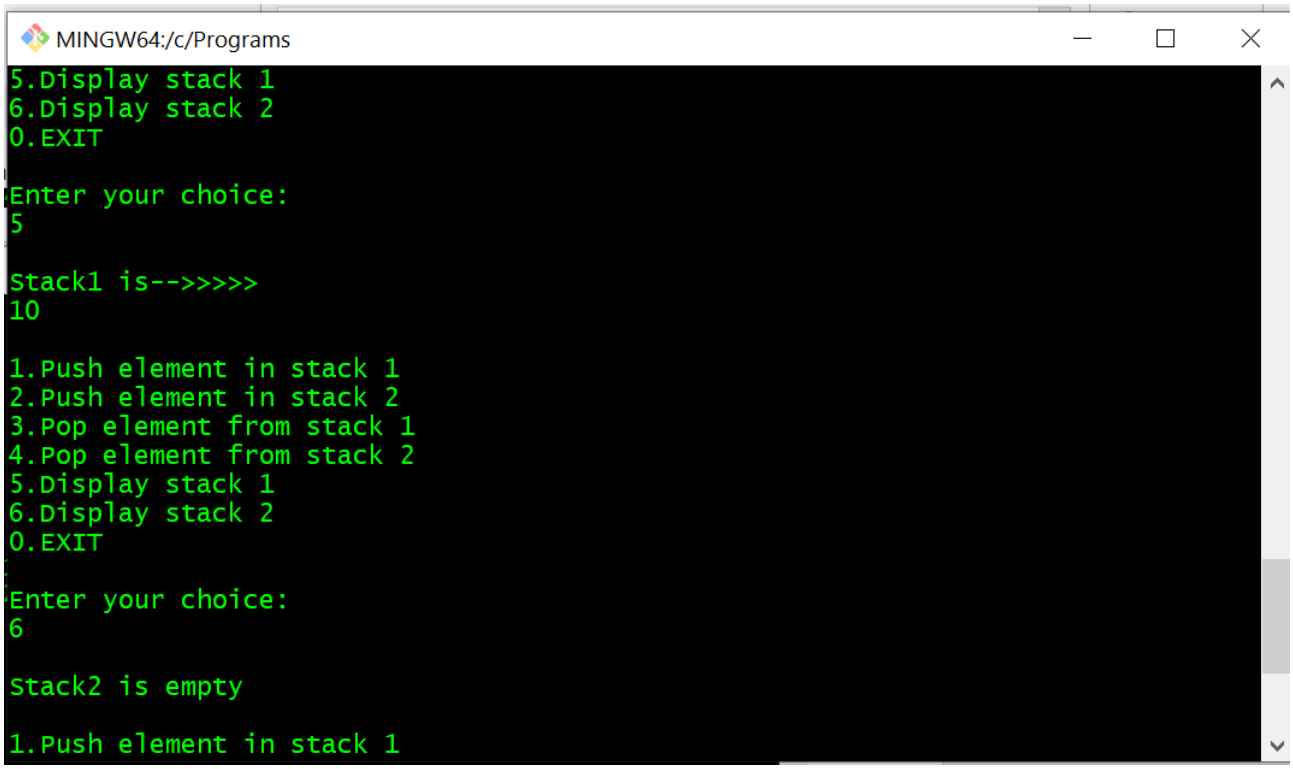
}

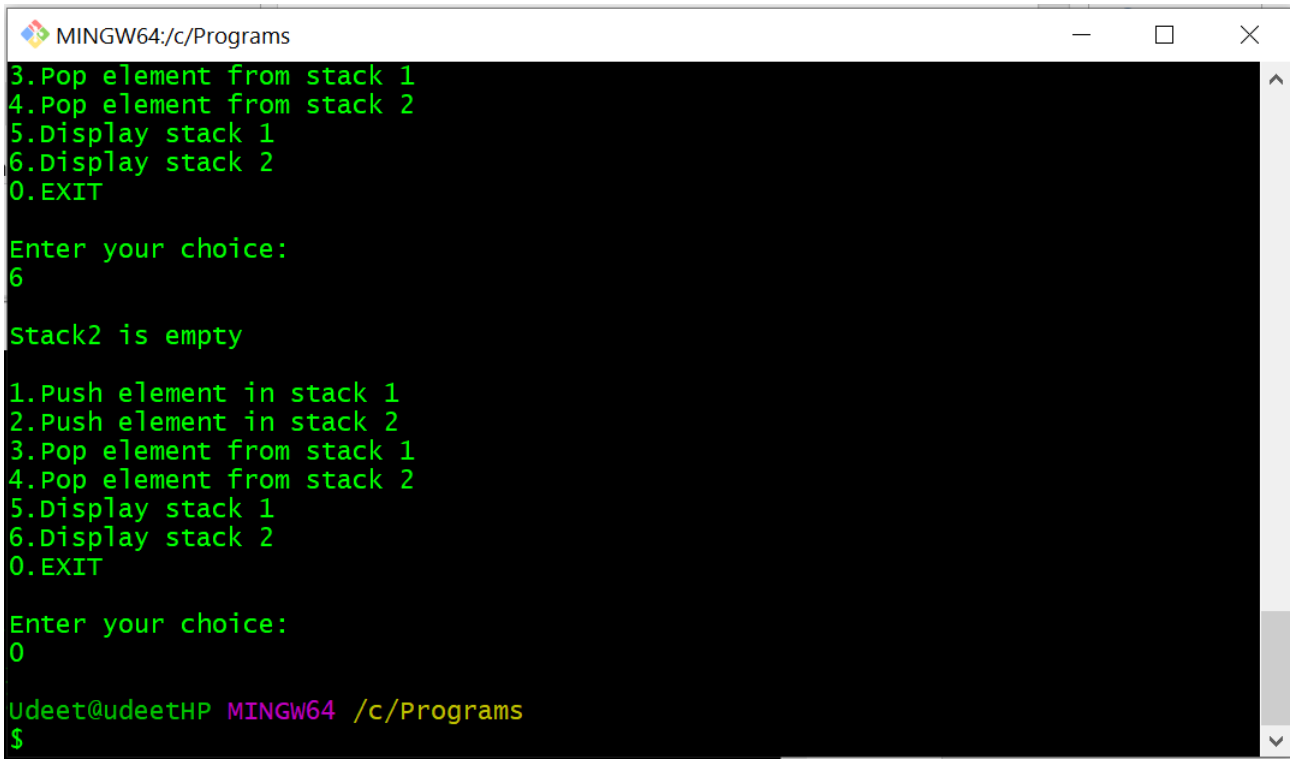

MINGW64:/c/Programs
Udeet@udeetHP MINGW64 /c/Programs
$ gcc q3.c
Udeet@udeetHP MINGW64 /c/Programs
$ ./a
Name: Udeet Mittal
Batch:C3
Roll Number: 64
Enter size of array you want to use:
3
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
1
Enter the element:


MINGW64:/c/Programs
Enter the element:
10
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
1
Enter the element:
20
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT


MINGW64:/c/Programs
Enter your choice:
1
Enter the element:
30
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
5
Stack1 is-->>>>>
30 20 10
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2


MINGW64:/c/Programs
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
3
Popped element is 30
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
3
Popped element is 20
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2


MINGW64:/c/Programs
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
5
Stack1 is-->>>>>
10
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
6
Stack2 is empty
1.Push element in stack 1


MINGW64:/c/Programs
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
6
Stack2 is empty
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
0.EXIT
Enter your choice:
0
Udeet@udeetHP MINGW64 /c/Programs
$