DS LAB
Lab 6

Q1)

Source Code:
“prefixeval.h”

```
#include <stdlib.h>
#include <math.h>
#include <string.h>
#define MAX 80

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

char *reverse(char *str)
{
    int len = strlen(str);
    for (int i = 0; i < len; i++)
    {
        rev[i] = str[len - i - 1];
    }
    return rev;
}
int isDigit(char ch)
{
    int c = (int)(ch);
    c = c - 48;
    // printf("%d", c);
    if (c >= 0 && c <= 9)
        return 1;
    else
        return 0;
}

int isOperand(char ch)
{
    switch (ch)
    {
    case '+':
    case '-':
    case '*':
    case '/':
        return 1;
        break;

    default:
        return 0;
        break;
    }
}

void push(struct stack *, int);
int stackFull(struct stack *);
double pop(struct stack *);
int stackEmpty(struct stack *);
double evalexpr(double, double, char);
void display(struct stack *);

void push(struct stack *Stack, int item)
{
    // printf("Item: %d\n", item);
    if (stackFull(Stack) == 1)
        exit(EXIT_FAILURE);
    Stack->items[++Stack->top] = item;
    // display(Stack);
}

int stackFull(struct stack *Stack)
{
    if (Stack->top == MAX)
    {
        return 1;
    }
    return 0;
}

double pop(struct stack *Stack)
{
    if (stackEmpty(Stack) == 1)
        exit(EXIT_FAILURE);
    double ele = Stack->items[Stack->top];
    Stack->top--;
    // display(Stack);
    return ele;
}

int stackEmpty(struct stack *Stack)
{
    if (Stack->top == -1)
        return 1;
    return 0;
}

double evalexpr(double op1, double op2, char opr)
{
    switch (opr)
    {
    case '+':
        return op1 + op2;
        break;
    case '-':
        return op1 - op2;
        break;
    case '*':
        return op1 * op2;
        break;
    case '/':
        return op1 - op2;
        break;
    default:
        break;
    }
}

void display(struct stack * Stack)
{
    for (int i = 0; i <= Stack->top ; i++)
    {
        printf("%lf\t", Stack->items[i]);
    }
    printf("\n");
}
```

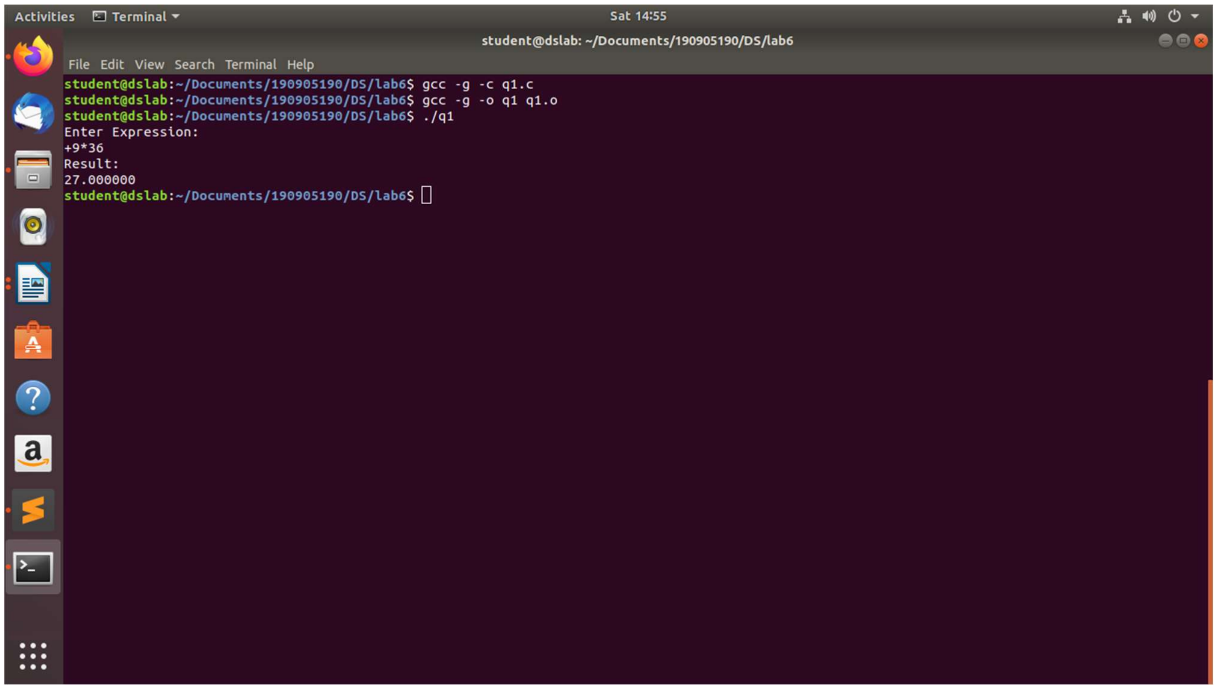
“q1.c”

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

int main()
{
    struct stack * Stack;
    Stack = malloc(sizeof(struct stack));
    Stack->top = -1;
    char str[MAX];
    printf("Enter Expression: \n");
    scanf("%s", str);
    char * rev;
    rev = reverse(str);
    // printf("%s\n", rev);
    for (int i = 0; i < strlen(rev); i++)
    {
        int is_digit = isDigit(rev[i]);
        int is_oper = isOperand(rev[i]);
        // printf("%d\t%d\n", is_digit, is_oper);
        if (is_digit == 1)
        {
            push(Stack, rev[i] - '0');
            // display(Stack);
        }
        else if(is_oper == 1)
        {
          double op1 = pop(Stack);
          // printf("OP1: %lf\n", op1);
          double op2 = pop(Stack);
          // printf("OP2: %lf\n", op2);
          double result = evalexpr(op1, op2, rev[i]);
          // printf("%lf\t%lf\t%lf\n", op1, op2, result);
          push(Stack, result);
        }


    }
    printf("Result: \n");
    display(Stack);
    return 0;
}
```

Output :


Activities
Terminal
Sat 14:55
student@dslab: ~/Documents/190905190/DS/lab6
File Edit View Search Terminal Help

```
student@dslab:~/Documents/190905190/DS/lab6$ gcc -g -c q1.c
student@dslab:~/Documents/190905190/DS/lab6$ gcc -g -o q1 q1.o
student@dslab:~/Documents/190905190/DS/lab6$ ./q1
Enter Expression:
+9*36
Result:
27.000000
student@dslab:~/Documents/190905190/DS/lab6$
```

Q2)

Source Code:
“infixtoprefix.h”

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 80

struct Stack {
   int top;
   int array[MAX];
};

struct Stack* create()
{
   struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
   stack->top = -1;
   return stack;
}

int isFull(struct Stack* stack)
{
   if(stack->top == MAX - 1){
      printf("Overflow\n");
   }

   return stack->top == MAX - 1;
}

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

void push(struct Stack* stack, int item)
{
   if (isFull(stack))
      return;
   stack->array[++stack->top] = item;
}

int pop(struct Stack* stack)
{
   if (isEmpty(stack))
      return INT_MIN;
   return stack->array[stack->top--];
}

int peek(struct Stack* stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top];
}

int checkIfOperand(char ch)
{
    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');
}

int precedence(char ch)
{
    switch (ch)
    {
    case '+':
    case '-':
        return 1;

    case '*':
    case '/':
        return 2;

    case '^':
        return 3;
    }
    return -1;
}


int getPostfix(char* expression)
{
    int i, j;
    struct Stack* stack = create();
    if(!stack)
        return -1 ;

    for (i = 0, j = -1; expression[i]; ++i)
    {
        if (checkIfOperand(expression[i]))
            expression[++j] = expression[i];

        else if (expression[i] == '(')
            push(stack, expression[i]);

        else if (expression[i] == ')')
        {
            while (!isEmpty(stack) && peek(stack) != '(')
                expression[++j] = pop(stack);
            if (!isEmpty(stack) && peek(stack) != '(')
                return -1;
            else
                pop(stack);
        }
        else
        {
            while (!isEmpty(stack) && precedence(expression[i]) <= precedence(peek(stack)))
                expression[++j] = pop(stack);
            push(stack, expression[i]);
        }

    }

    while (!isEmpty(stack))
        expression[++j] = pop(stack);

    expression[++j] = '\0';

}

void reverse(char *exp){

    int size = strlen(exp);
    int j = size, i=0;
    char temp[size];

    temp[j--]='\0';
    while(exp[i]!='\0')
    {
        temp[j] = exp[i];
        j--;
        i++;
    }
    strcpy(exp,temp);
}
void brackets(char* exp){
    int i = 0;
    while(exp[i]!='\0')
    {
        if(exp[i]=='(')
            exp[i]=')';
        else if(exp[i]==')')
            exp[i]='(';
        i++;
    }
}
void InfixtoPrefix(char *exp){

    int size = strlen(exp);

    reverse(exp);

    brackets(exp);

    getPostfix(exp);
    reverse(exp);
}
```

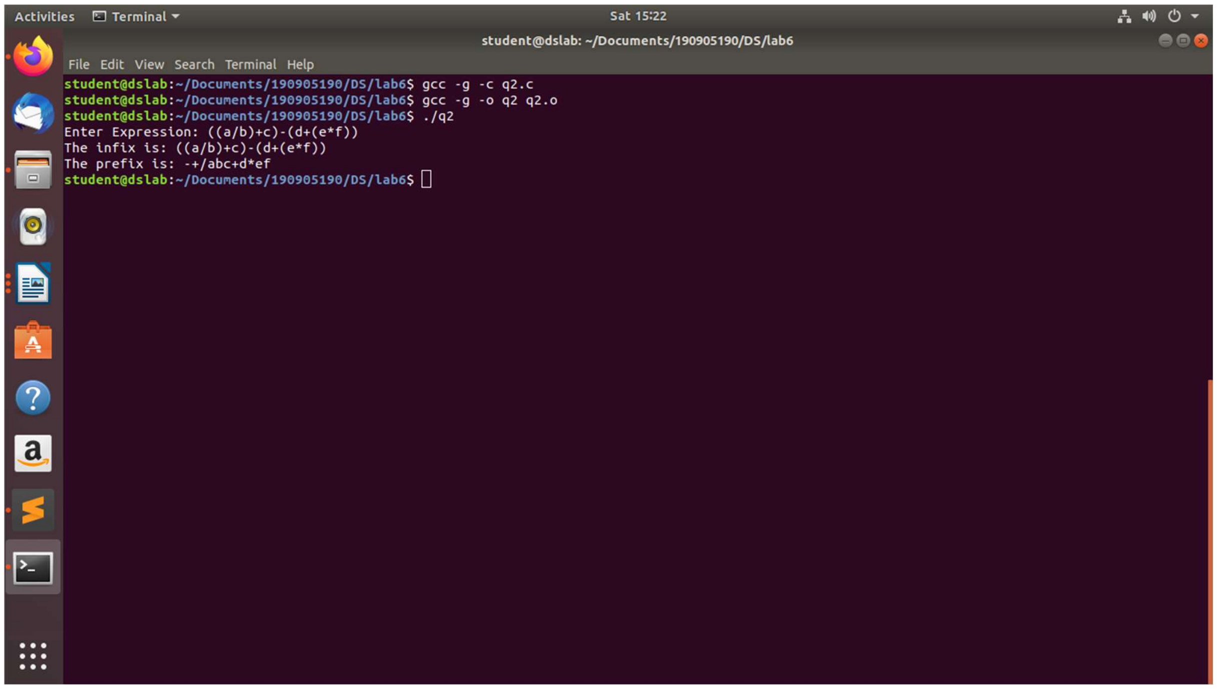
“q2.c”

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "infixtoprefix.h"

int main()
{
    char expression[80];
    printf("Enter Expression: ");
    scanf("%s", expression);
    printf("The infix is: ");
    printf("%s\n",expression);
    InfixtoPrefix(expression);
    printf("The prefix is: ");
    printf("%s\n",expression);

    return 0;
}
```

Output:


Activities
Terminal
Sat 15:22
student@dslab: ~/Documents/190905190/DS/lab6
File Edit View Search Terminal Help

```
student@dslab:~/Documents/190905190/DS/lab6$ gcc -g -c q2.c
student@dslab:~/Documents/190905190/DS/lab6$ gcc -g -o q2 q2.o
student@dslab:~/Documents/190905190/DS/lab6$ ./q2
Enter Expression: ((a/b)+c)-(d+(e*f))
The infix is: ((a/b)+c)-(d+(e*f))
The prefix is: -+/abc+d*ef
student@dslab:~/Documents/190905190/DS/lab6$
```

Q3)
Source Code:
“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 stack {
   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* s = (stack *)malloc(sizeof(stack));
	s->top = -1;
	return s;
}
```

“q3.c”

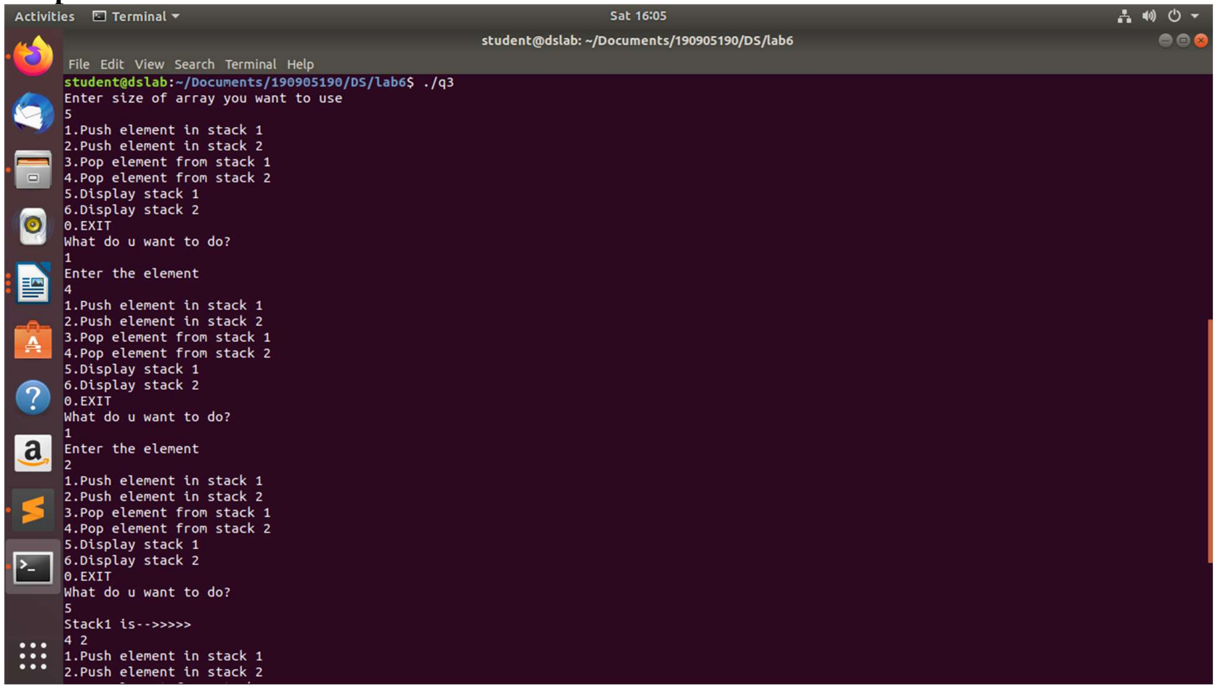
```
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"

void main()
{
	int n,top1,top2,ch=1,a,i,arr[100];
	printf("Enter size of array you want to use\n");
	scanf("%d",&n);top1=-1;
	top2=n;
	while(ch!=0)
	{
		printf("1.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("What do u want to do?\n");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
			{
				printf("Enter the element\n");
				scanf("%d",&a);
				if(top1!=(top2-1))
					arr[++top1]=a;
				else
					printf("Overflow\n");break;
			}
			case 2:
```

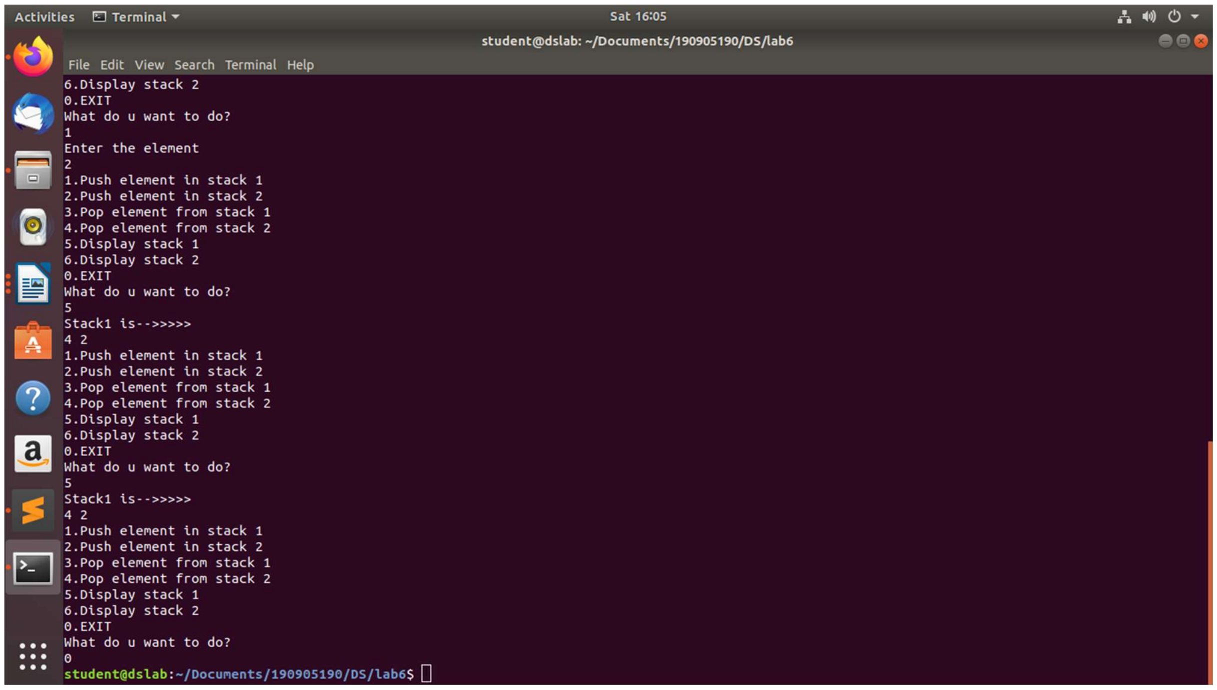
```
{
        printf("Enter the element\n");
        scanf("%d",&a);
        if(top2!=(top1+1))
                arr[--top2]=a;
        else
                printf("Overflow\n");
        break;
}
case 3:
{
        if(top1==-1)
                printf("Stack1 is empty\n");
        else
        {
                a=arr[top1--];
                printf("%d\n",a);
        }
break;
}
case 4:
{
        if(top2==n)
                printf("Stack2 is empty\n");
        else
        {
                a=arr[top2++];
                printf("%d\n",a);
        }
break;
}
case 5:
{
        if(top1==-1)
                printf("Stack1 is empty\n");
        else
        {
                printf("Stack1 is-->>>>>\n");
                for(i=0;i<=top1;i++)
                printf("%d ",arr[i]);
                printf("\n");
        }
break;
}
case 6:
{
        if(top2==n)
                printf("Stack2 is empty\n");
        else
        {
                printf("Stack2 is-->>>>>>\n");
                for(i=(n-1);i>=top2;i--)
```

```
                                        printf("%d ",arr[i]);
                                        printf("\n");
                                }
                        break;
                        }
                        case 0:
                        break;
                }
}
}
```

Output:



```
student@dslab:~/Documents/190905190/DS/lab6$ ./q3
Enter size of array you want to use
5
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
What do u want to do?
1
Enter the element
4
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
What do u want to do?
1
Enter the element
2
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
What do u want to do?
5
Stack1 is-->>>>>
4 2
1.Push element in stack 1
2.Push element in stack 2
```


Activities
Terminal
Sat 16:05
student@dslab: ~/Documents/190905190/DS/lab6
File Edit View Search Terminal Help

```
6.Display stack 2
0.EXIT
What do u want to do?
1
Enter the element
2
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
What do u want to do?
5
Stack1 is-->>>>>
4 2
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
What do u want to do?
5
Stack1 is-->>>>>
4 2
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
What do u want to do?
0
student@dslab:~/Documents/190905190/DS/lab6$
```