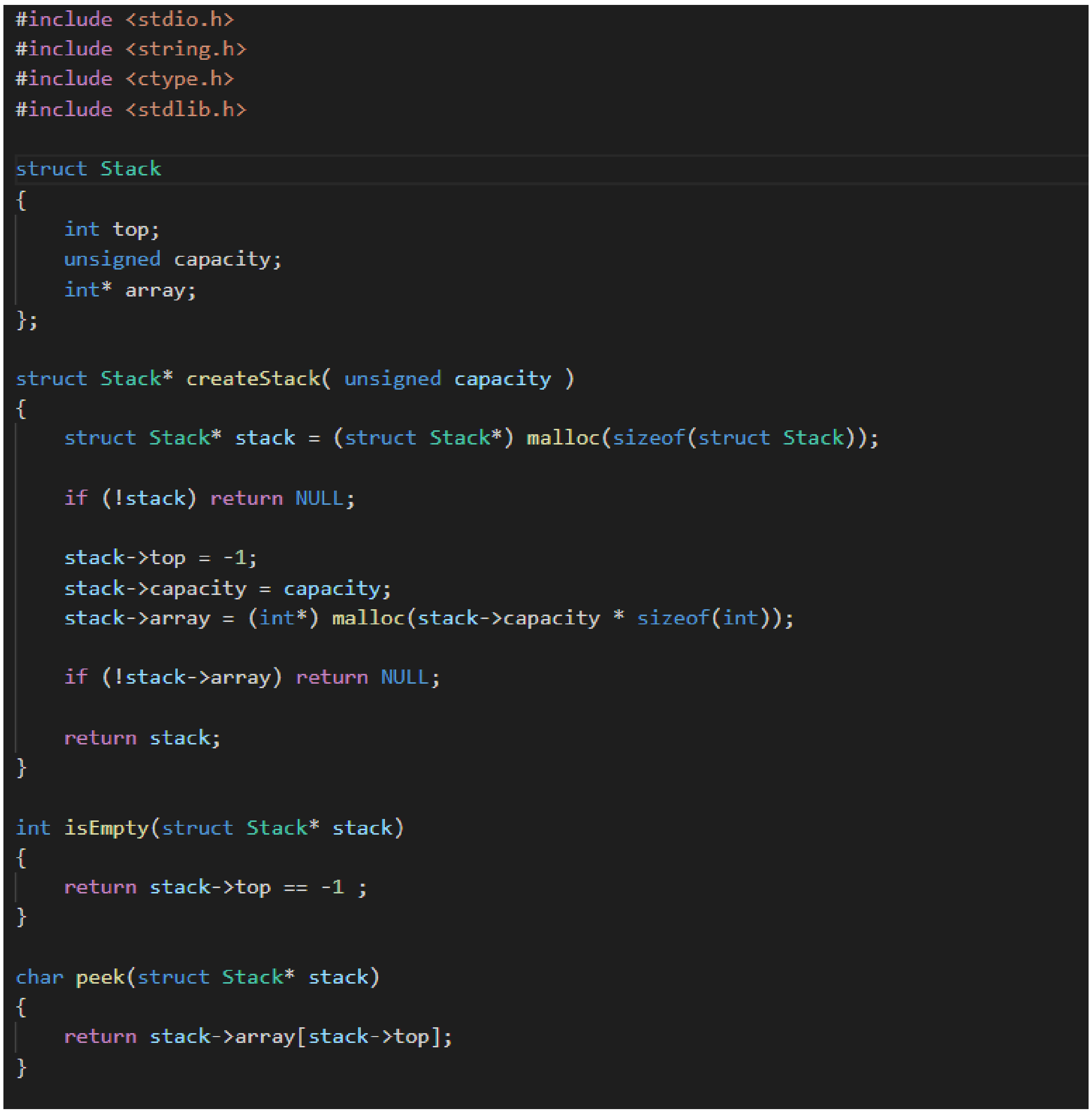
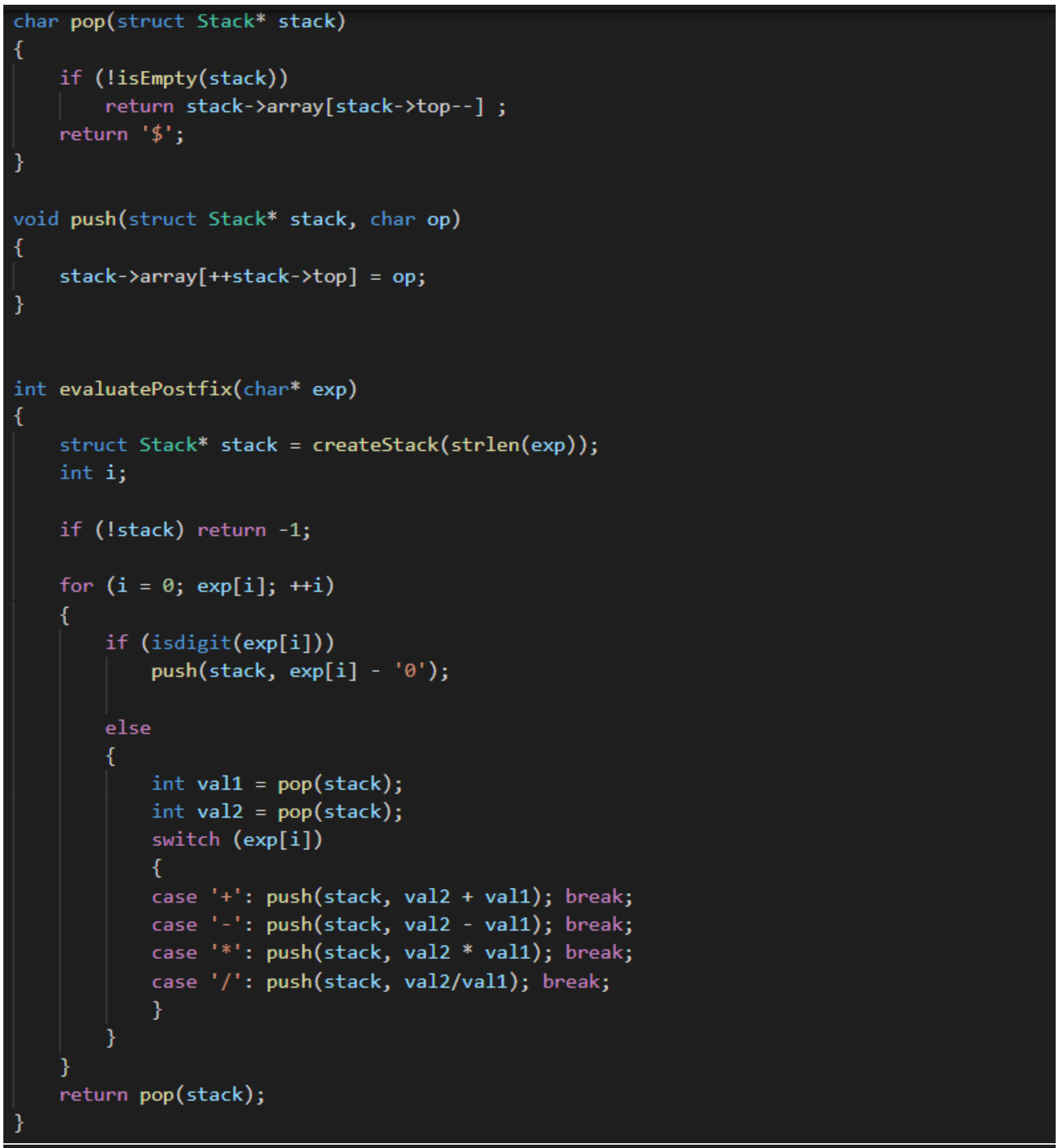
Agnim Gupta
2028083
A23
CSSE

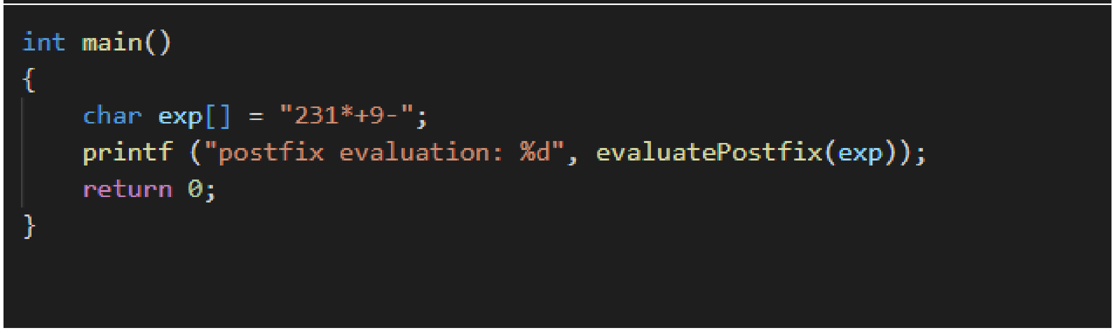
Question 1



```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

struct Stack
{
    int top;
    unsigned capacity;
    int* array;
};

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];
}
```

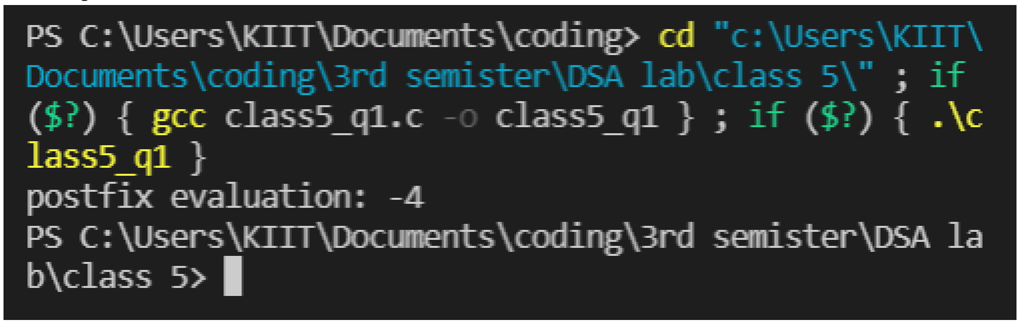


```c
char pop(struct Stack* stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--] ;
    return '$';
}

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);
}
```



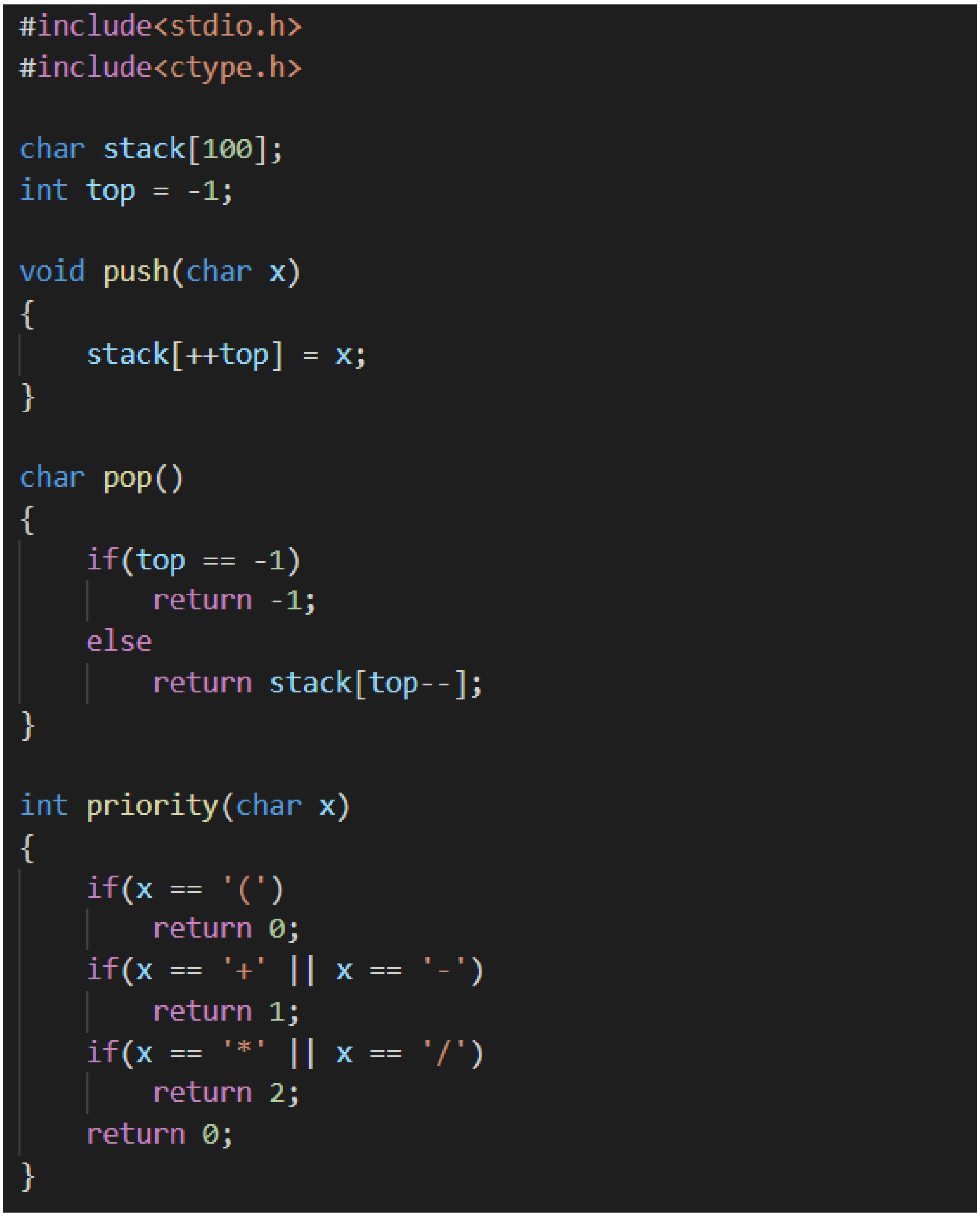
```c
int main()
{
    char exp[] = "231*+9-";
    printf ("postfix evaluation: %d", evaluatePostfix(exp));
    return 0;
}
```

Output

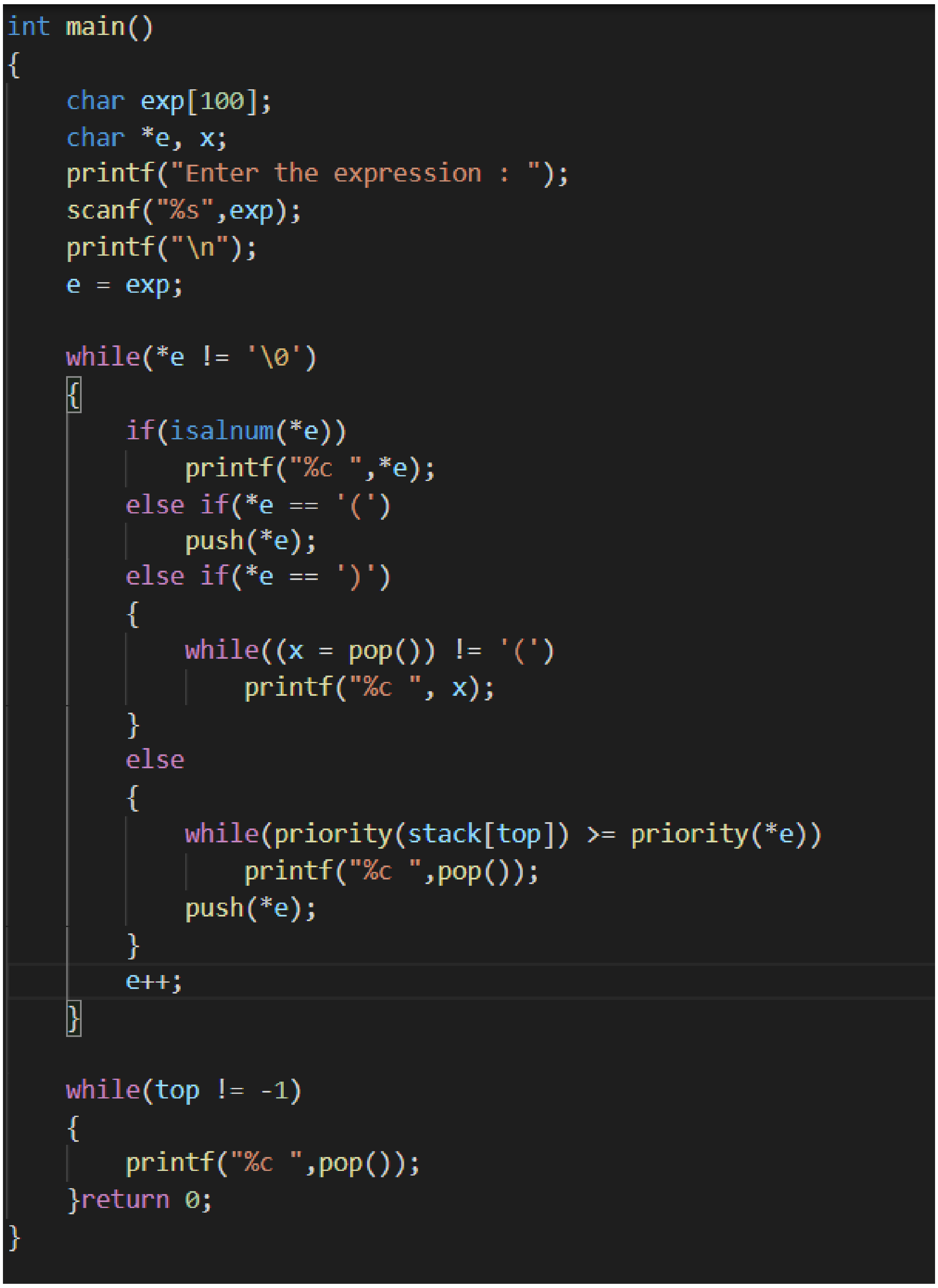


```
PS C:\Users\KIIT\Documents\coding> cd "c:\Users\KIIT\Documents\coding\3rd semister\DSA lab\class 5\" ; if ($?) { gcc class5_q1.c -o class5_q1 } ; if ($?) { .\class5_q1 }
postfix evaluation: -4
PS C:\Users\KIIT\Documents\coding\3rd semister\DSA lab\class 5>
```

Question 2

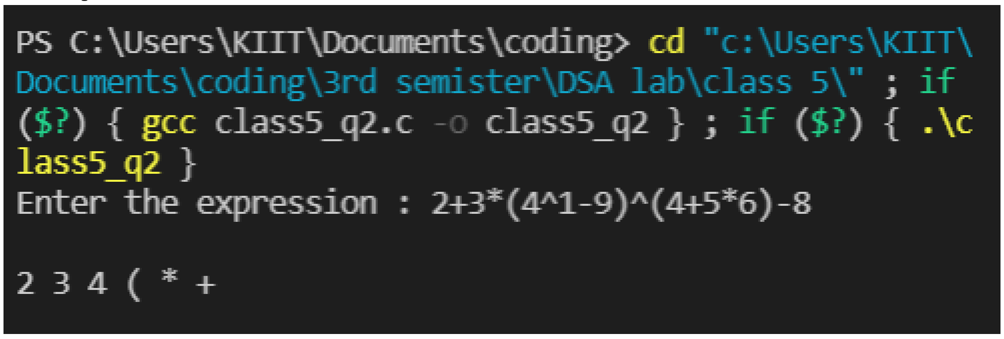


```c
#include<stdio.h>
#include<ctype.h>

char stack[100];
int top = -1;

void push(char x)
{
    stack[++top] = x;
}

char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}
```



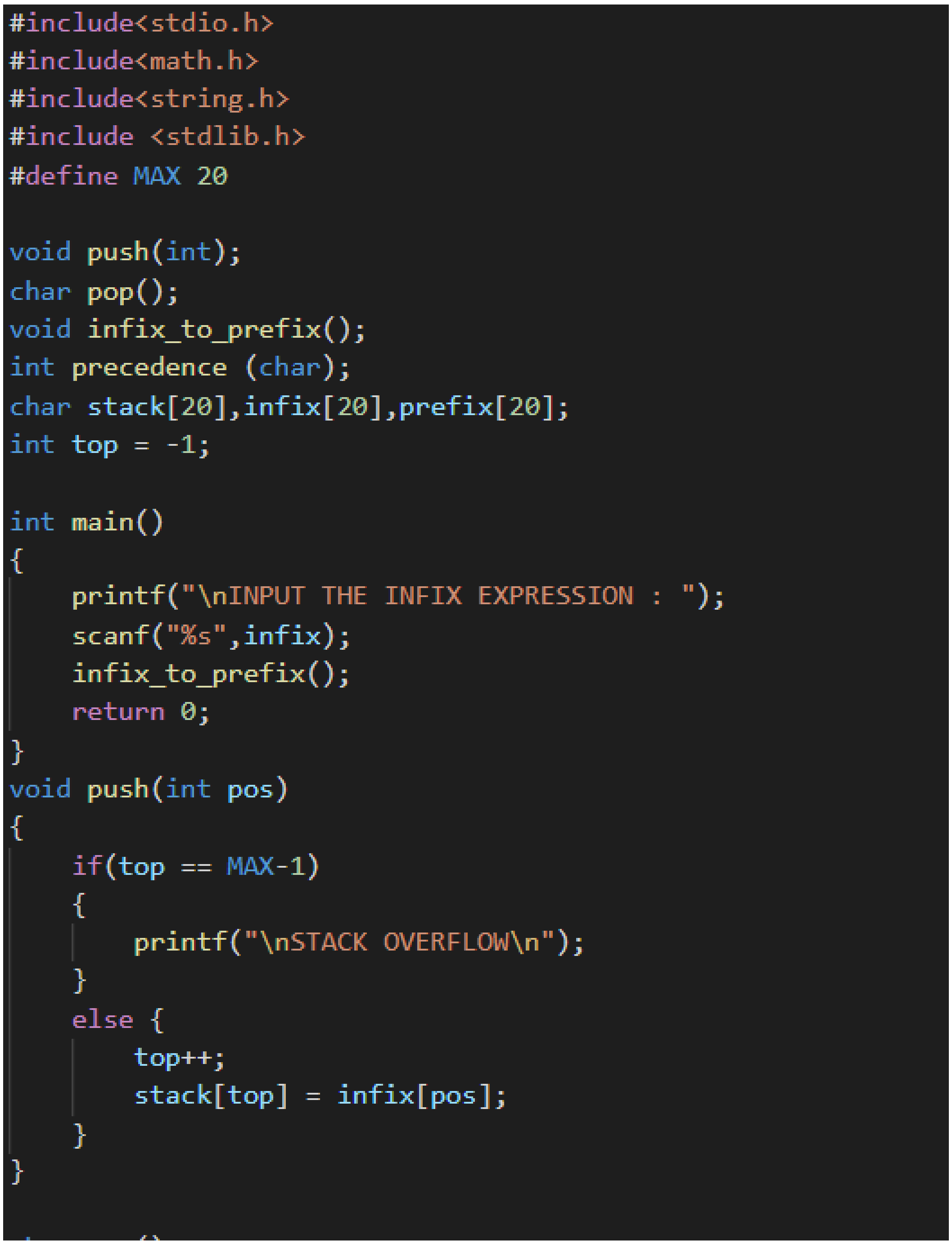
```c
int main()
{
    char exp[100];
    char *e, x;
    printf("Enter the expression : ");
    scanf("%s",exp);
    printf("\n");
    e = exp;

    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c ",*e);
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((x = pop()) != '(')
                printf("%c ", x);
        }
        else
        {
            while(priority(stack[top]) >= priority(*e))
                printf("%c ",pop());
            push(*e);
        }
        e++;
    }

    while(top != -1)
    {
        printf("%c ",pop());
    }return 0;
}
```

Output

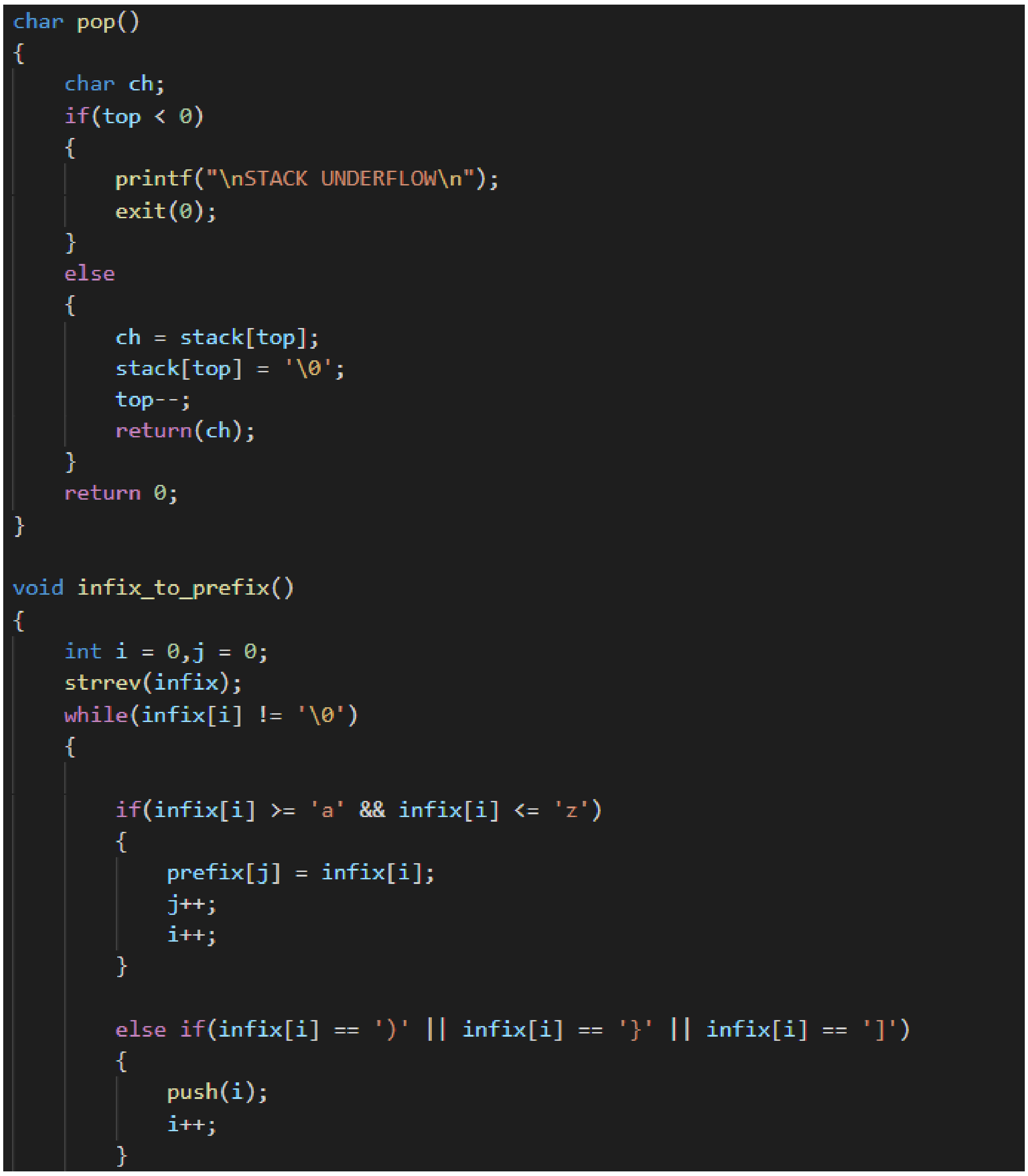


```
PS C:\Users\KIIT\Documents\coding> cd "c:\Users\KIIT\Documents\coding\3rd semister\DSA lab\class 5\" ; if ($?) { gcc class5_q2.c -o class5_q2 } ; if ($?) { .\class5_q2 }
Enter the expression : 2+3*(4^1-9)^(4+5*6)-8

2 3 4 ( * +
```

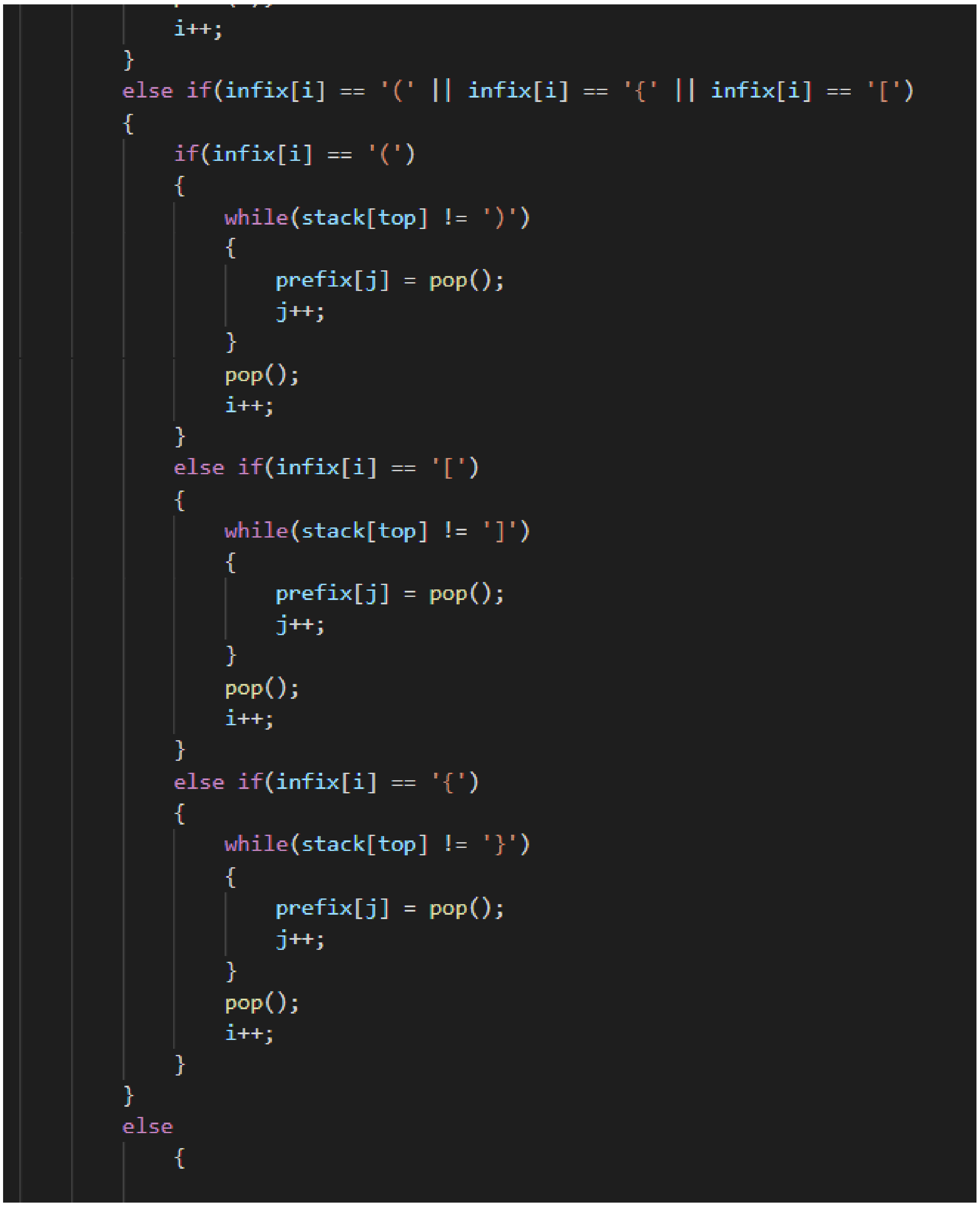
Question 3



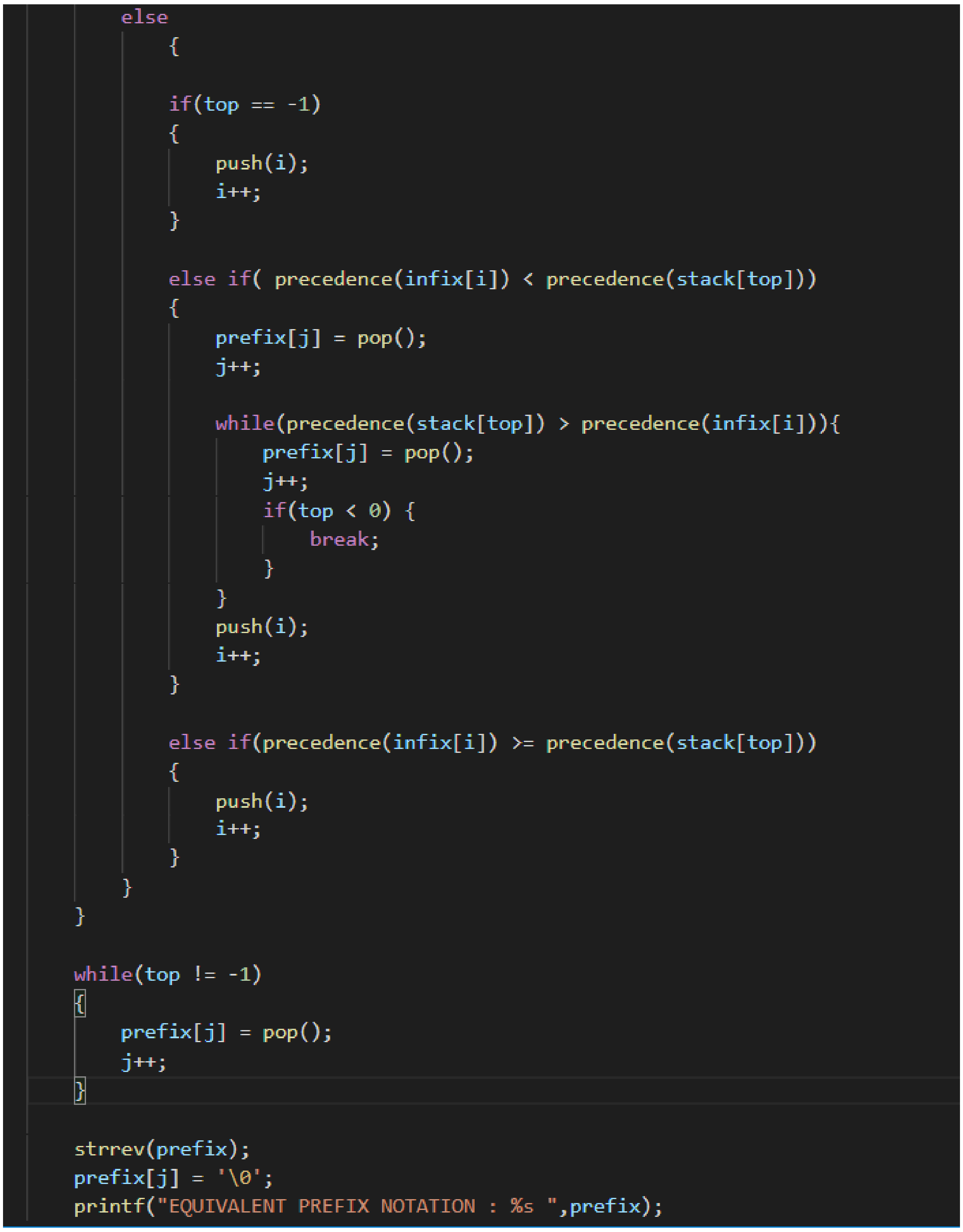
```c
#include<stdio.h>
#include<math.h>
#include<string.h>
#include <stdlib.h>
#define MAX 20

void push(int);
char pop();
void infix_to_prefix();
int precedence (char);
char stack[20],infix[20],prefix[20];
int top = -1;

int main()
{
    printf("\nINPUT THE INFIX EXPRESSION : ");
    scanf("%s",infix);
    infix_to_prefix();
    return 0;
}
void push(int pos)
{
    if(top == MAX-1)
    {
        printf("\nSTACK OVERFLOW\n");
    }
    else {
        top++;
        stack[top] = infix[pos];
    }
}
```



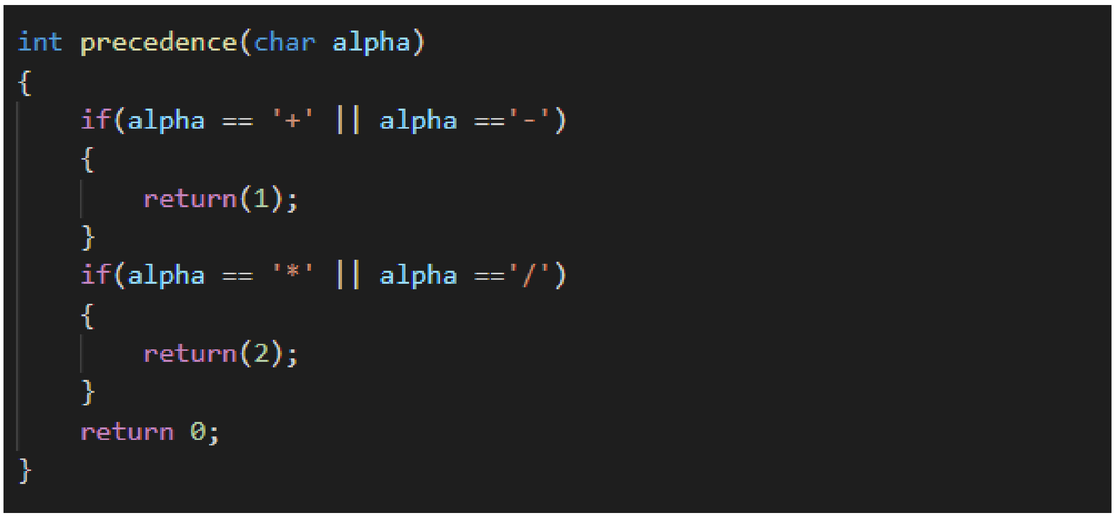
```
char pop()
{
    char ch;
    if(top < 0)
    {
        printf("\nSTACK UNDERFLOW\n");
        exit(0);
    }
    else
    {
        ch = stack[top];
        stack[top] = '\0';
        top--;
        return(ch);
    }
    return 0;
}

void infix_to_prefix()
{
    int i = 0,j = 0;
    strrev(infix);
    while(infix[i] != '\0')
    {

        if(infix[i] >= 'a' && infix[i] <= 'z')
        {
            prefix[j] = infix[i];
            j++;
            i++;
        }

        else if(infix[i] == ')' || infix[i] == '}' || infix[i] == ']')
        {
            push(i);
            i++;
        }
```



```
                i++;
            }
            else if(infix[i] == '(' || infix[i] == '{' || infix[i] == '[')
            {
                if(infix[i] == '(')
                {
                    while(stack[top] != ')')
                    {
                        prefix[j] = pop();
                        j++;
                    }
                    pop();
                    i++;
                }
                else if(infix[i] == '[')
                {
                    while(stack[top] != ']')
                    {
                        prefix[j] = pop();
                        j++;
                    }
                    pop();
                    i++;
                }
                else if(infix[i] == '{')
                {
                    while(stack[top] != '}')
                    {
                        prefix[j] = pop();
                        j++;
                    }
                    pop();
                    i++;
                }
            }
            else
                {
```

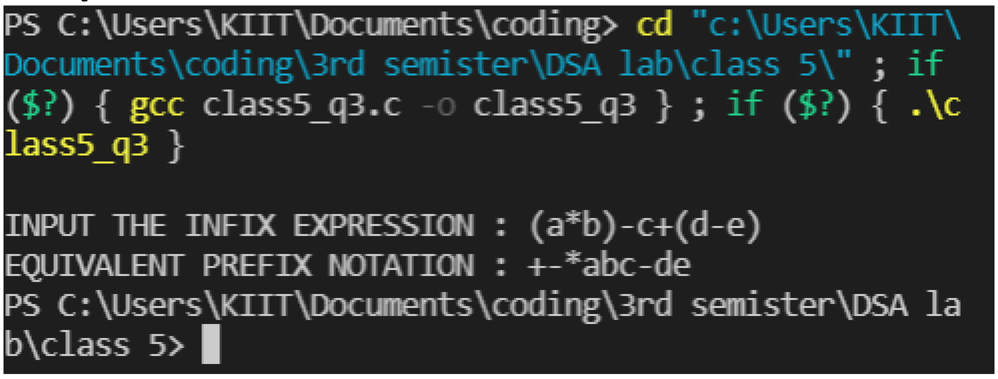


```
        else
            {

            if(top == -1)
            {
                push(i);
                i++;
            }

            else if( precedence(infix[i]) < precedence(stack[top]))
            {
                prefix[j] = pop();
                j++;

                while(precedence(stack[top]) > precedence(infix[i])){
                    prefix[j] = pop();
                    j++;
                    if(top < 0) {
                        break;
                    }
                }
                push(i);
                i++;
            }

            else if(precedence(infix[i]) >= precedence(stack[top]))
            {
                push(i);
                i++;
            }
        }
    }

    while(top != -1)
    {
        prefix[j] = pop();
        j++;
    }

    strrev(prefix);
    prefix[j] = '\0';
    printf("EQUIVALENT PREFIX NOTATION : %s ",prefix);
```



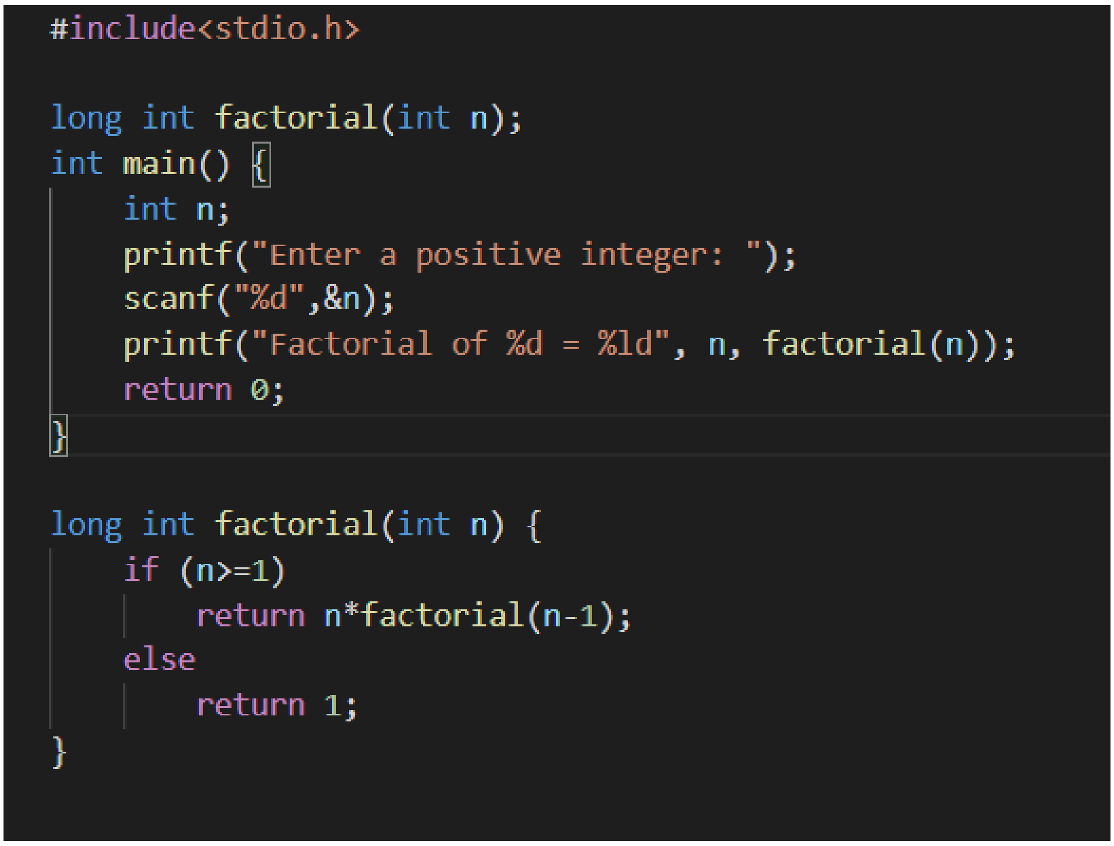
```
int precedence(char alpha)
{
    if(alpha == '+' || alpha =='-')
    {
        return(1);
    }
    if(alpha == '*' || alpha =='/')
    {
        return(2);
    }
    return 0;
}
```

Output



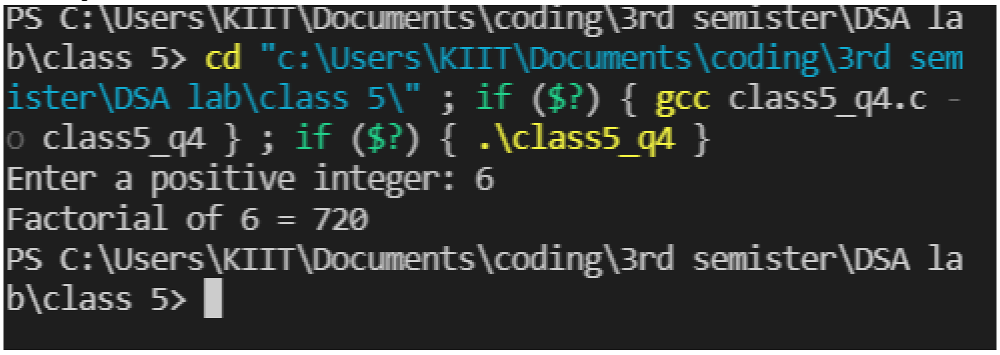
```
PS C:\Users\KIIT\Documents\coding> cd "c:\Users\KIIT\
Documents\coding\3rd semister\DSA lab\class 5\" ; if
($?) { gcc class5_q3.c -o class5_q3 } ; if ($?) { .\c
lass5_q3 }

INPUT THE INFIX EXPRESSION : (a*b)-c+(d-e)
EQUIVALENT PREFIX NOTATION : +-*abc-de
PS C:\Users\KIIT\Documents\coding\3rd semister\DSA la
b\class 5> 
```

Question 4



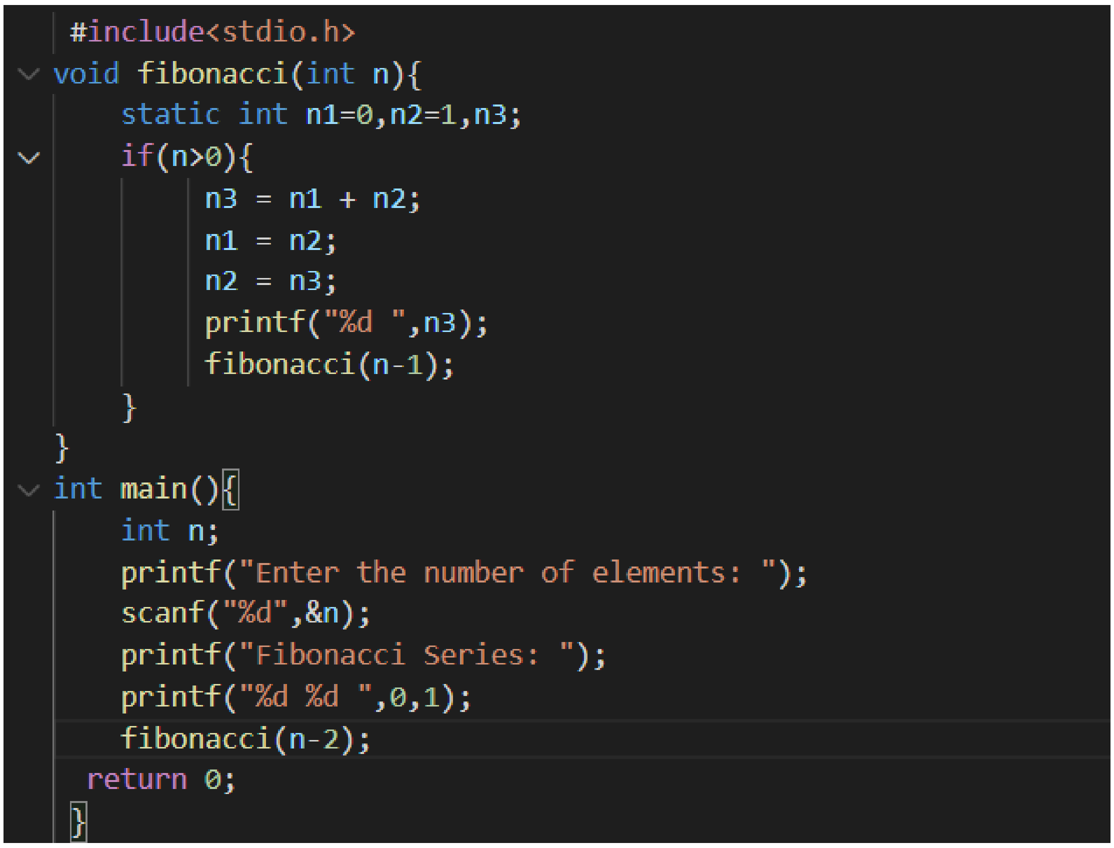
```c
#include<stdio.h>

long int factorial(int n);
int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    printf("Factorial of %d = %ld", n, factorial(n));
    return 0;
}

long int factorial(int n) {
    if (n>=1)
        return n*factorial(n-1);
    else
        return 1;
}
```

Output



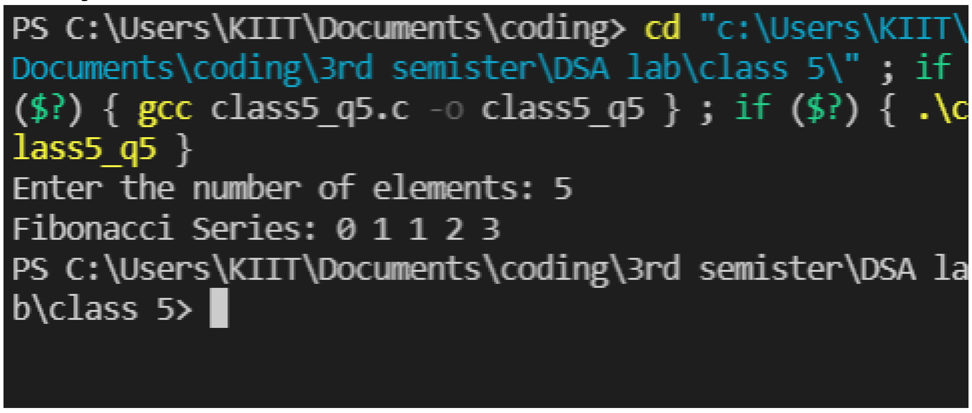
```
PS C:\Users\KIIT\Documents\coding\3rd semister\DSA la
b\class 5> cd "c:\Users\KIIT\Documents\coding\3rd sem
ister\DSA lab\class 5\" ; if ($?) { gcc class5_q4.c -
o class5_q4 } ; if ($?) { .\class5_q4 }
Enter a positive integer: 6
Factorial of 6 = 720
PS C:\Users\KIIT\Documents\coding\3rd semister\DSA la
b\class 5> 
```

Question 5



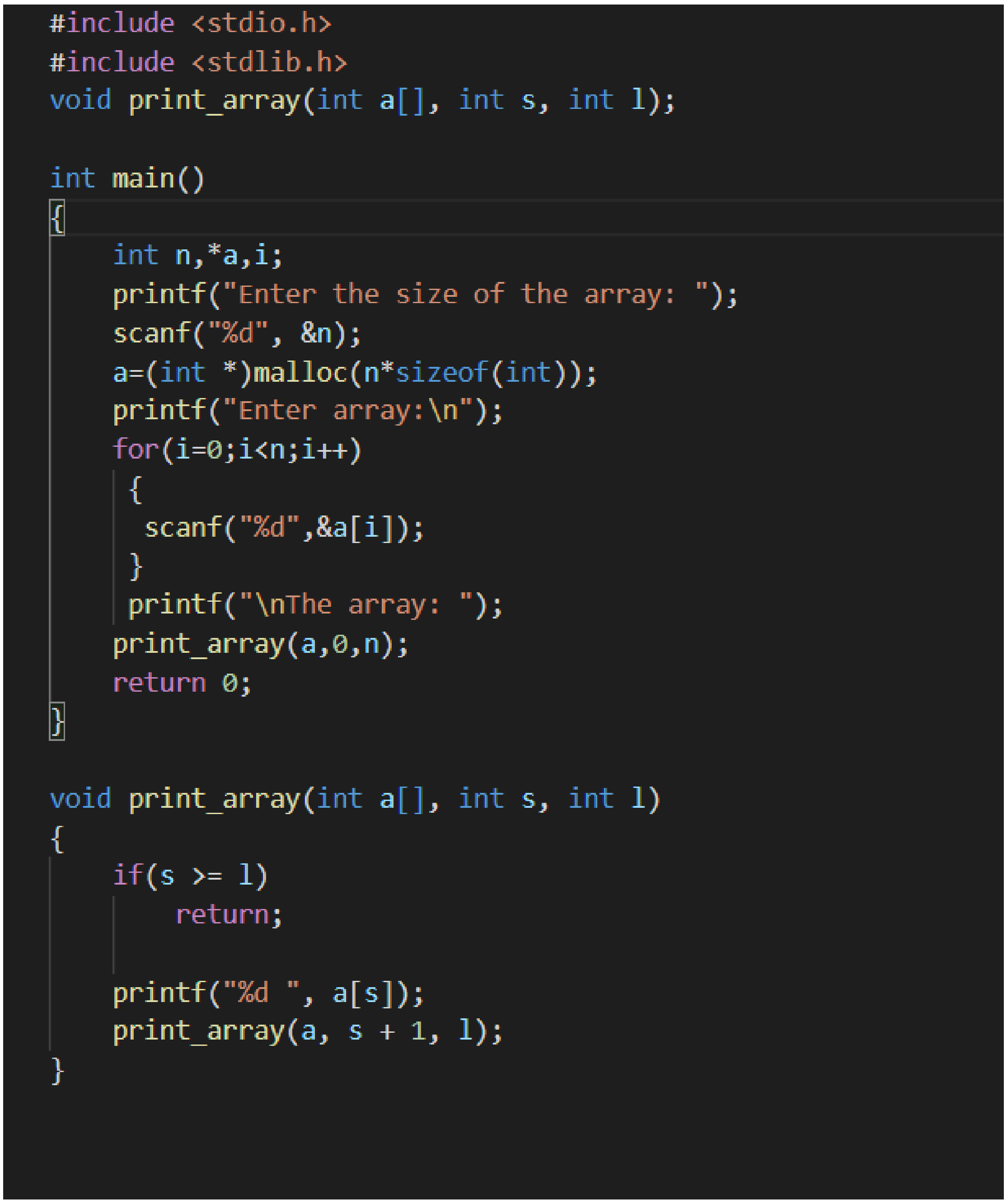
```c
#include<stdio.h>
void fibonacci(int n){
    static int n1=0,n2=1,n3;
    if(n>0){
        n3 = n1 + n2;
        n1 = n2;
        n2 = n3;
        printf("%d ",n3);
        fibonacci(n-1);
    }
}
int main(){
    int n;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Fibonacci Series: ");
    printf("%d %d ",0,1);
    fibonacci(n-2);
  return 0;
}
```

Output



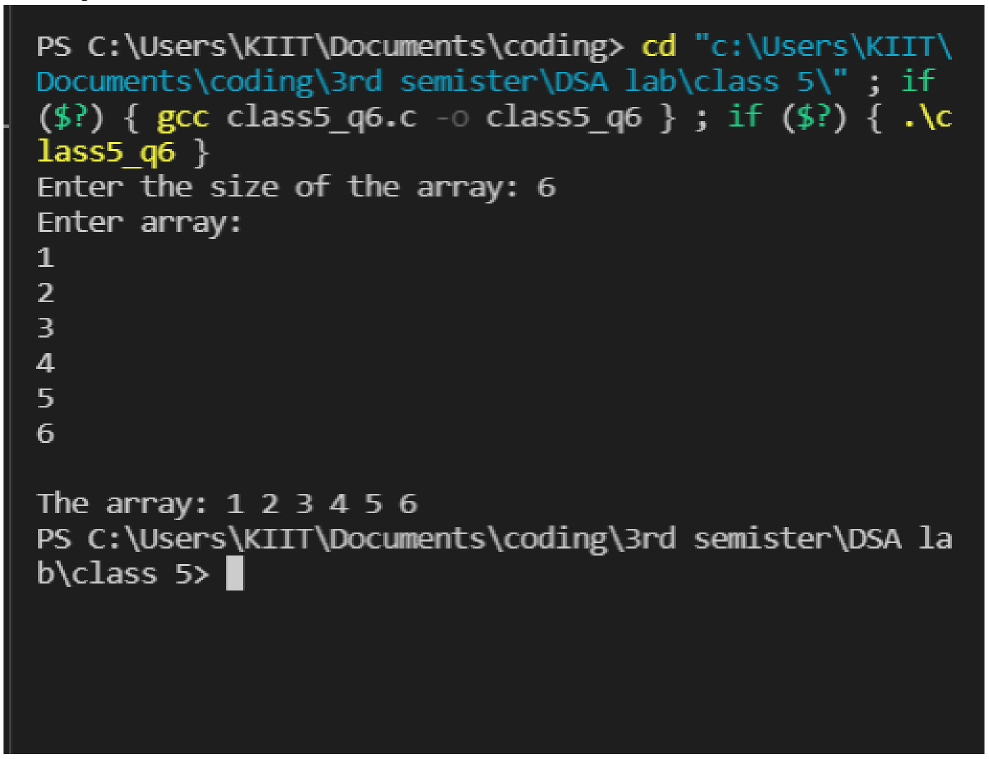
```
PS C:\Users\KIIT\Documents\coding> cd "c:\Users\KIIT\
Documents\coding\3rd semister\DSA lab\class 5\" ; if
($?) { gcc class5_q5.c -o class5_q5 } ; if ($?) { .\c
lass5_q5 }
Enter the number of elements: 5
Fibonacci Series: 0 1 1 2 3
PS C:\Users\KIIT\Documents\coding\3rd semister\DSA la
b\class 5>
```

Question 6



```c
#include <stdio.h>
#include <stdlib.h>
void print_array(int a[], int s, int l);

int main()
{
    int n,*a,i;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    a=(int *)malloc(n*sizeof(int));
    printf("Enter array:\n");
    for(i=0;i<n;i++)
     {
      scanf("%d",&a[i]);
     }
     printf("\nThe array: ");
    print_array(a,0,n);
    return 0;
}

void print_array(int a[], int s, int l)
{
    if(s >= l)
        return;

    printf("%d ", a[s]);
    print_array(a, s + 1, l);
}
```

Output



```
PS C:\Users\KIIT\Documents\coding> cd "c:\Users\KIIT\
Documents\coding\3rd semister\DSA lab\class 5\" ; if
($?) { gcc class5_q6.c -o class5_q6 } ; if ($?) { .\c
lass5_q6 }
Enter the size of the array: 6
Enter array:
1
2
3
4
5
6

The array: 1 2 3 4 5 6
PS C:\Users\KIIT\Documents\coding\3rd semister\DSA la
b\class 5>
```