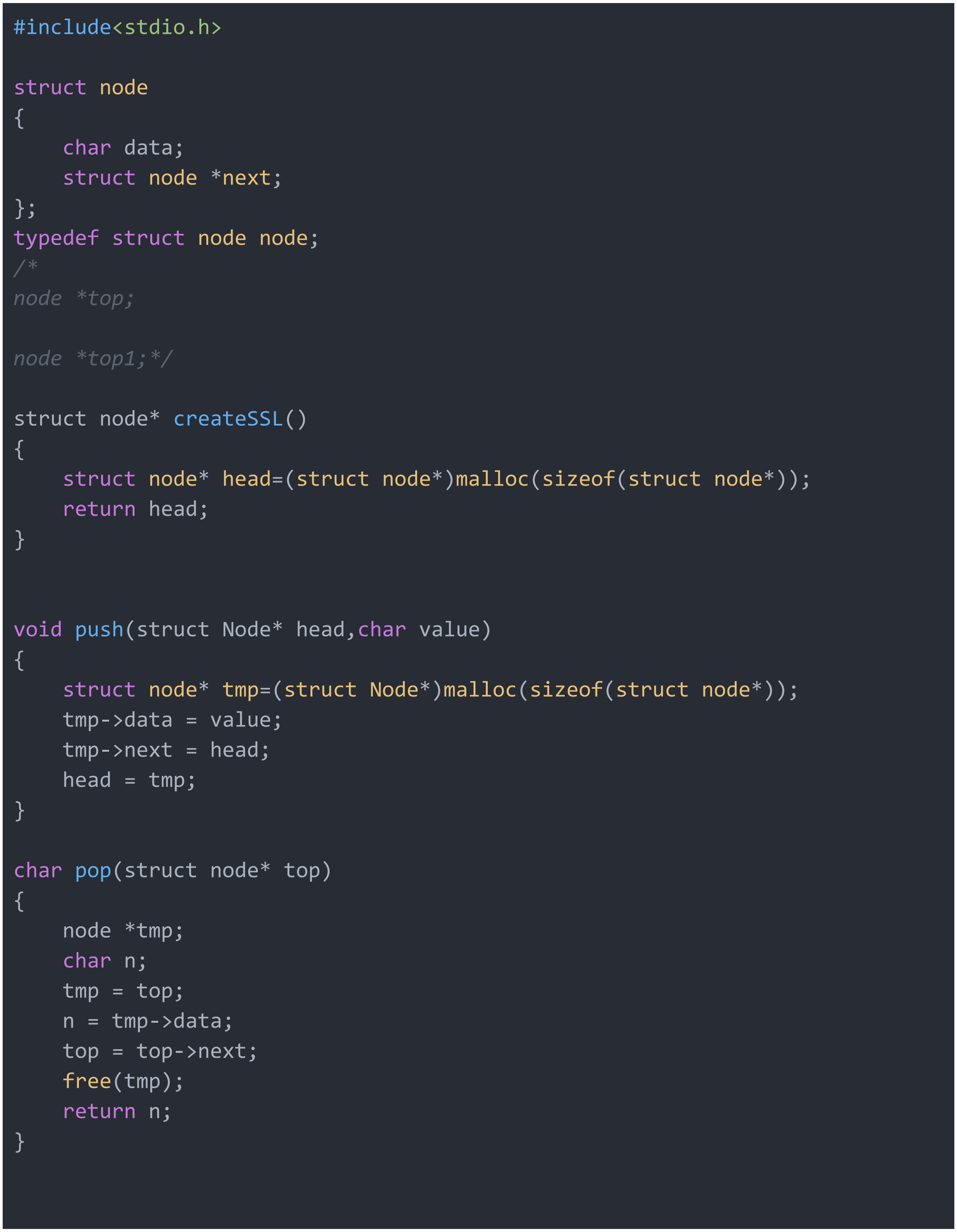
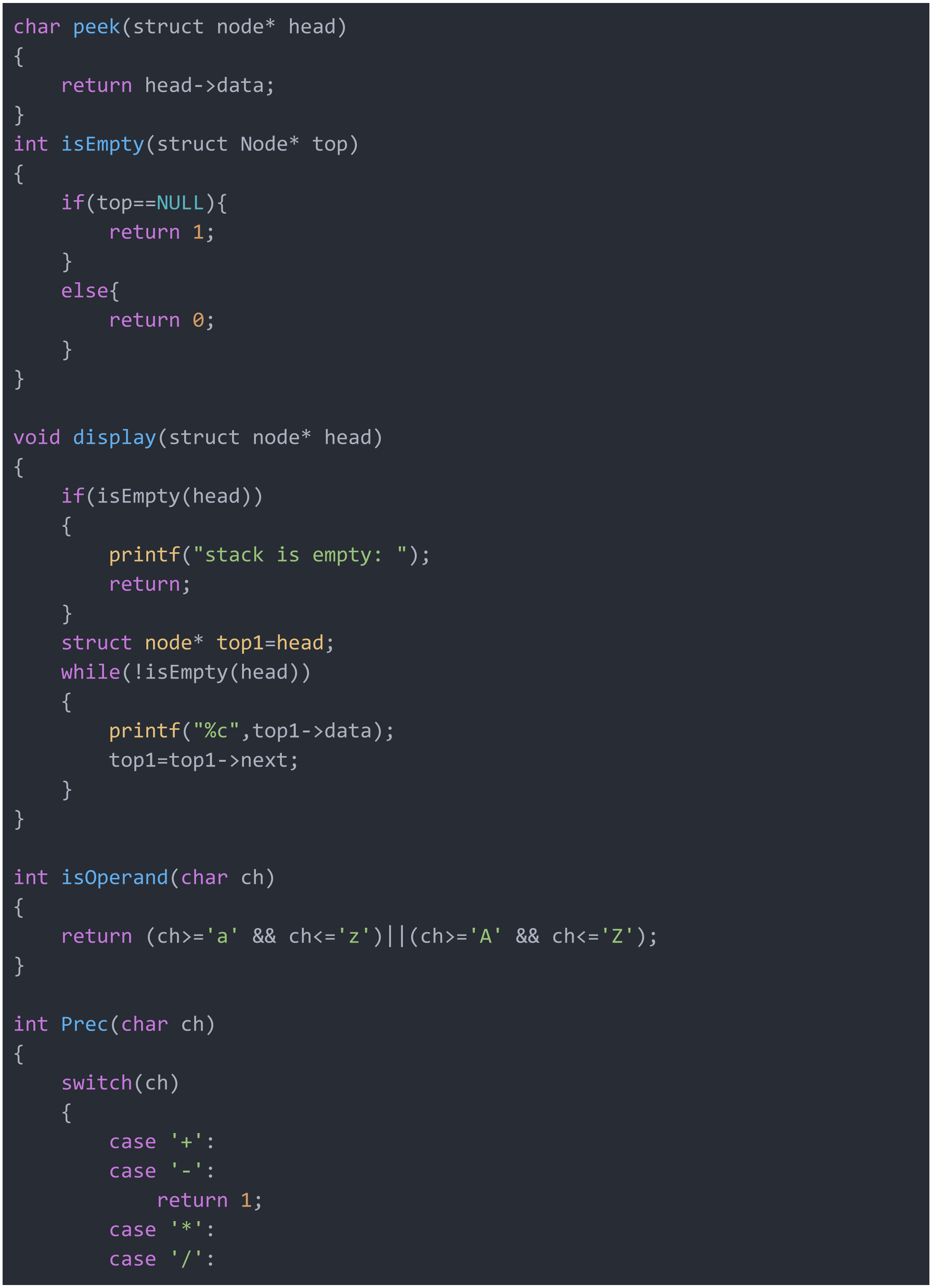
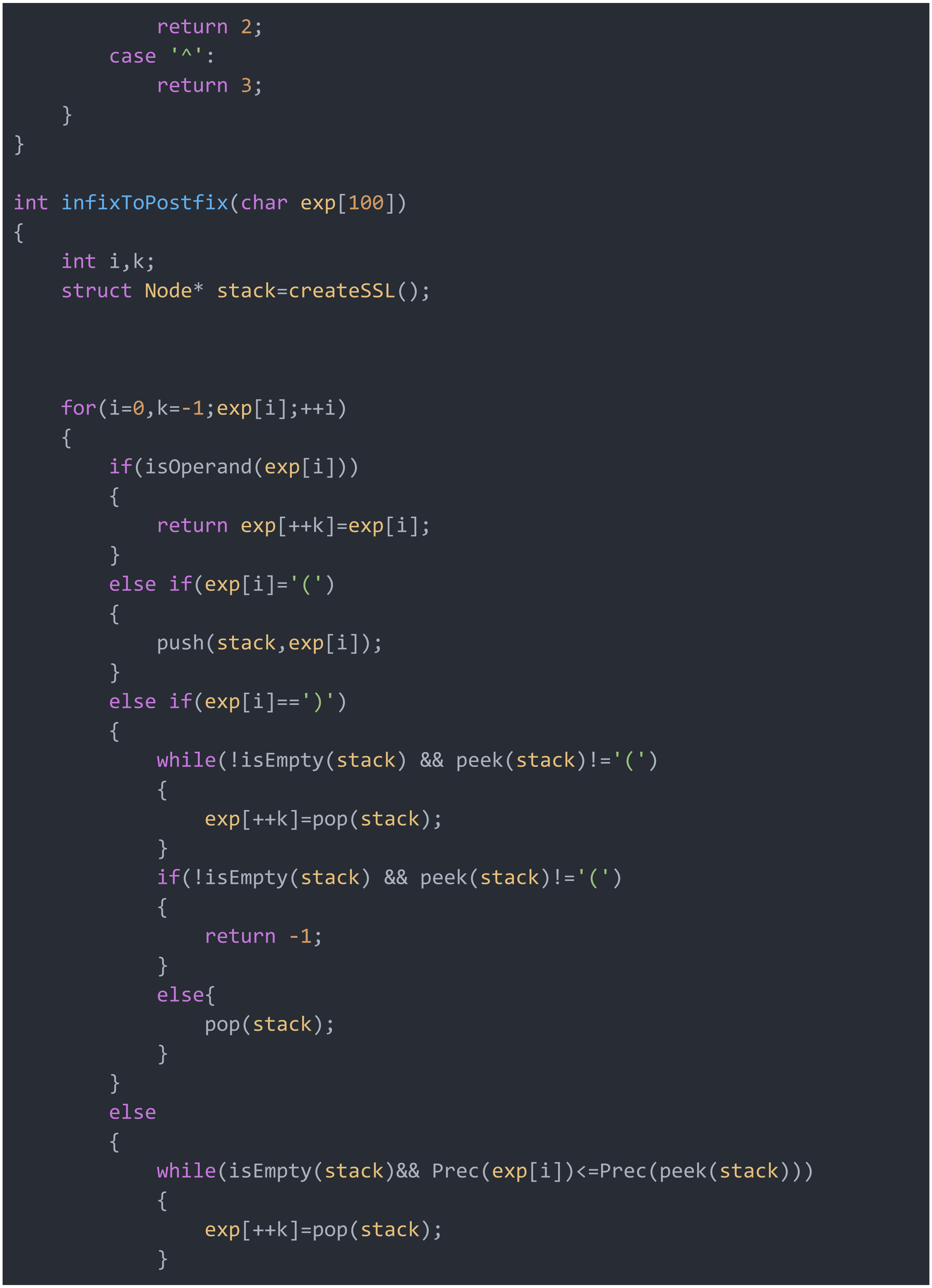
Name- Arya Nair
Roll Number- 16010421063



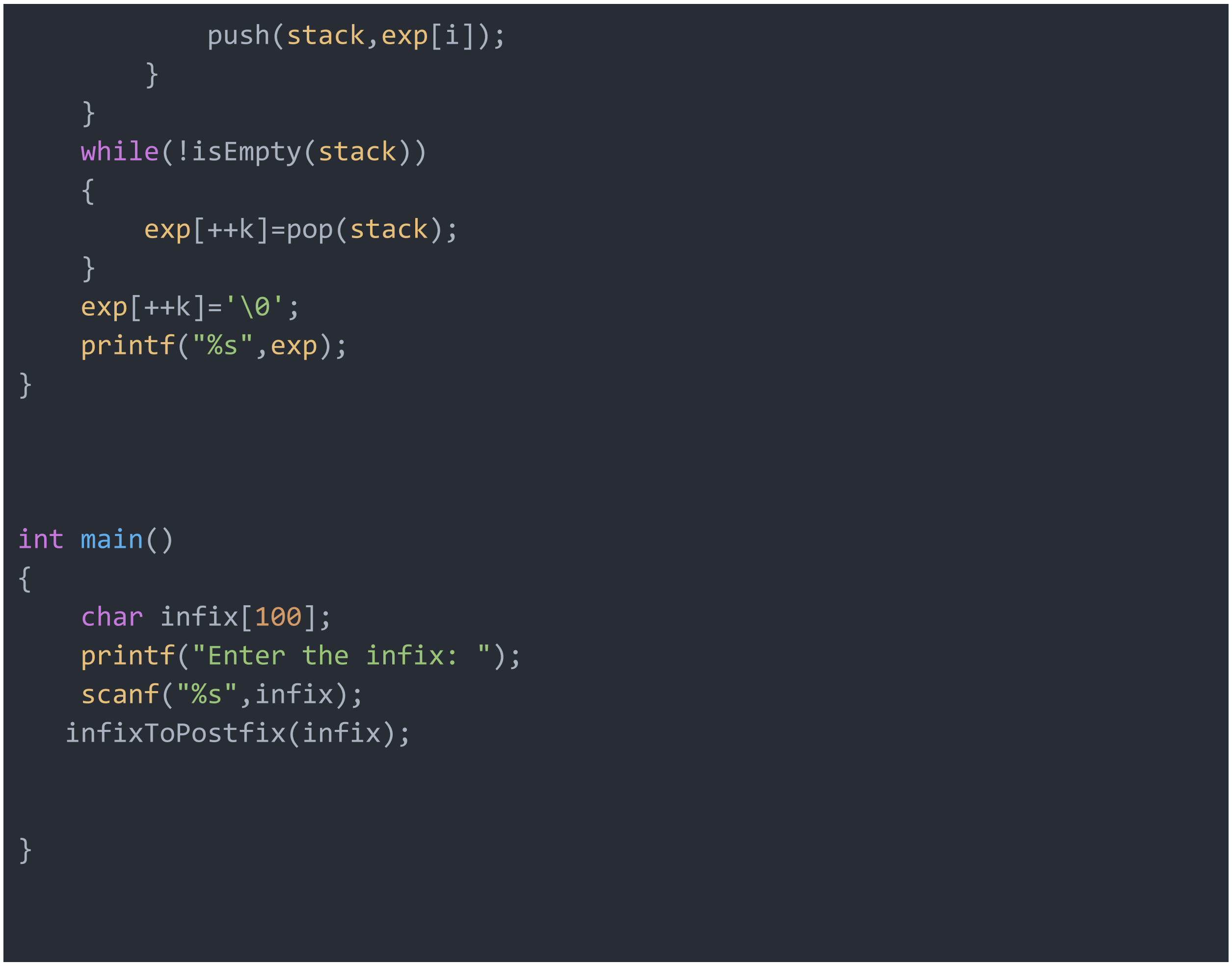
```
#include<stdio.h>

struct node
{
    char data;
    struct node *next;
};
typedef struct node node;
/*
node *top;

node *top1;*/

struct node* createSSL()
{
    struct node* head=(struct node*)malloc(sizeof(struct node*));
    return head;
}


void push(struct Node* head,char value)
{
    struct node* tmp=(struct Node*)malloc(sizeof(struct node*));
    tmp->data = value;
    tmp->next = head;
    head = tmp;
}

char pop(struct node* top)
{
    node *tmp;
    char n;
    tmp = top;
    n = tmp->data;
    top = top->next;
    free(tmp);
    return n;
}
```



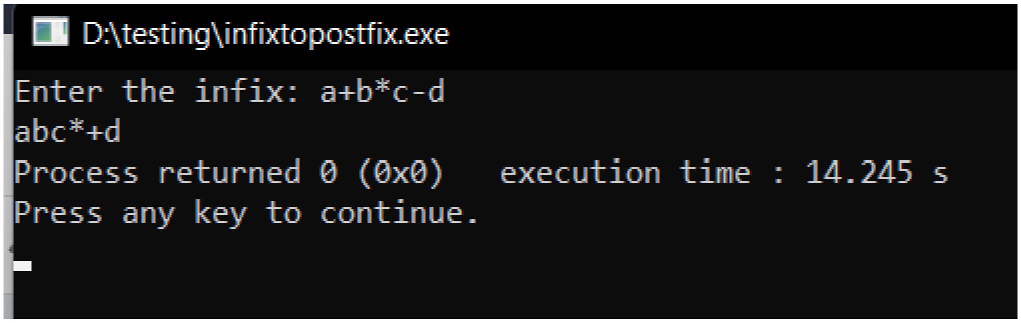
```
char peek(struct node* head)
{
    return head->data;
}
int isEmpty(struct Node* top)
{
    if(top==NULL){
        return 1;
    }
    else{
        return 0;
    }
}

void display(struct node* head)
{
    if(isEmpty(head))
    {
        printf("stack is empty: ");
        return;
    }
    struct node* top1=head;
    while(!isEmpty(head))
    {
        printf("%c",top1->data);
        top1=top1->next;
    }
}

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

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



```
            return 2;
        case '^':
            return 3;
    }
}

int infixToPostfix(char exp[100])
{
    int i,k;
    struct Node* stack=createSSL();



    for(i=0,k=-1;exp[i];++i)
    {
        if(isOperand(exp[i]))
        {
            return exp[++k]=exp[i];
        }
        else if(exp[i]='(')
        {
            push(stack,exp[i]);
        }
        else if(exp[i]==')')
        {
            while(!isEmpty(stack) && peek(stack)!='(')
            {
                exp[++k]=pop(stack);
            }
            if(!isEmpty(stack) && peek(stack)!='(')
            {
                return -1;
            }
            else{
                pop(stack);
            }
        }
        else
        {
            while(isEmpty(stack)&& Prec(exp[i])<=Prec(peek(stack)))
            {
                exp[++k]=pop(stack);
            }
```



```c
            push(stack,exp[i]);
        }
    }
    while(!isEmpty(stack))
    {
        exp[++k]=pop(stack);
    }
    exp[++k]='\0';
    printf("%s",exp);
}



int main()
{
    char infix[100];
    printf("Enter the infix: ");
    scanf("%s",infix);
   infixToPostfix(infix);


}
```



D:\testing\infixtopostfix.exe

```
Enter the infix: a+b*c-d
abc*+d
Process returned 0 (0x0)   execution time : 14.245 s
Press any key to continue.
```