CSE 2003

DATA STRUCTURES AND ALGORITHMS

Lab CAT

September 26, 2020

L19+L20 | SJT317

FALL SEMESTER 2020-21

by

SHARADINDU ADHIKARI

19BCE2105

Question 1

Problem:

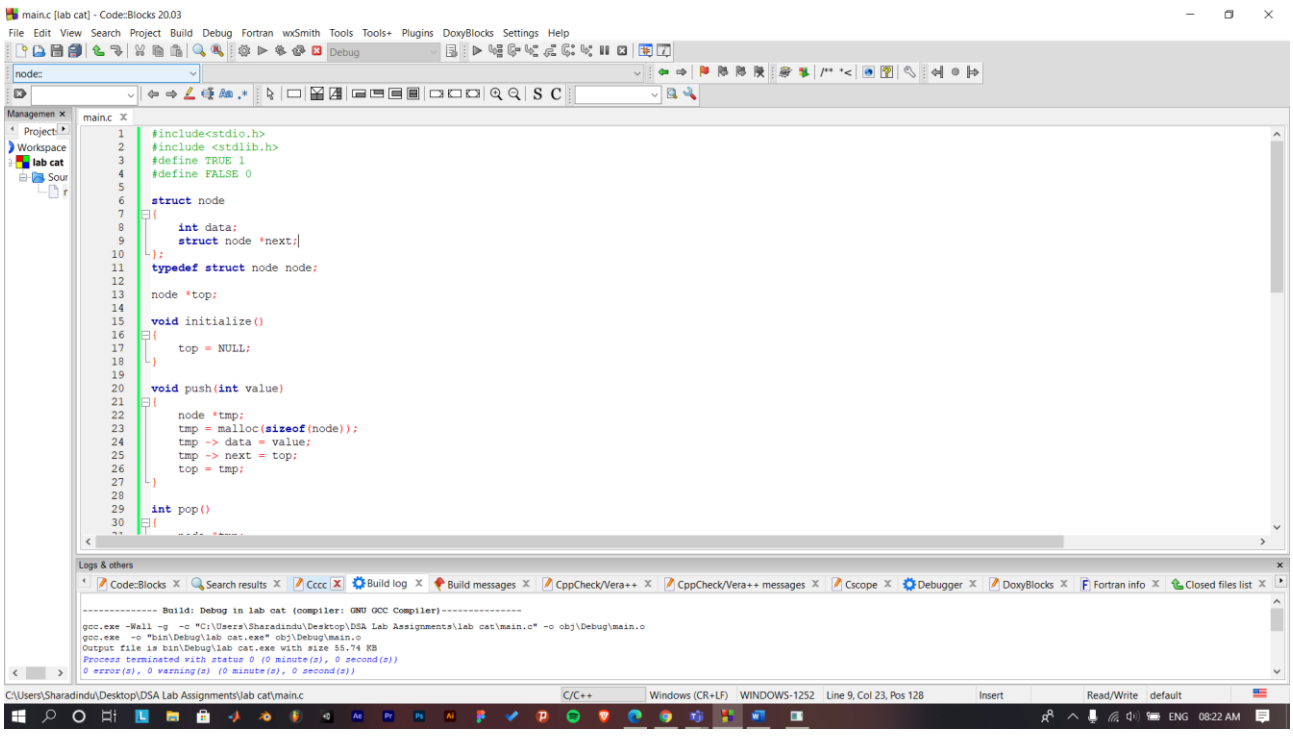
Write a program to implement a stack using an array and Linked List.

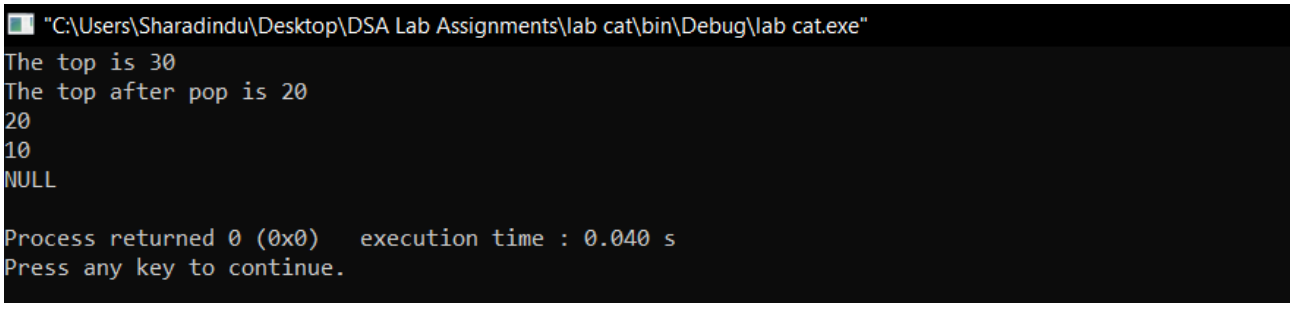
Code, SS & CMD:

Using Linked list:

```
#include<stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0

struct node
{
    int data;
    struct node *next;
};
typedef struct node node;

node *top;

void initialize()
{
    top = NULL;
}

void push(int value)
{
    node *tmp;
    tmp = malloc(sizeof(node));
    tmp -> data = value;
    tmp -> next = top;
    top = tmp;
}

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

int Top()
{
    return top->data;
}

int isempty()
{
    return top==NULL;
}
```

```c
void display(node *head)
{
    if(head == NULL)
    {
        printf("NULL\n");
    }
    else
    {
        printf("%d\n", head -> data);
        display(head->next);
    }
}

int main()
{
    initialize();
    push(10);
    push(20);
    push(30);
    printf("The top is %d\n",Top());
    pop();
    printf("The top after pop is %d\n",Top());
    display(top);
    return 0;
}
```

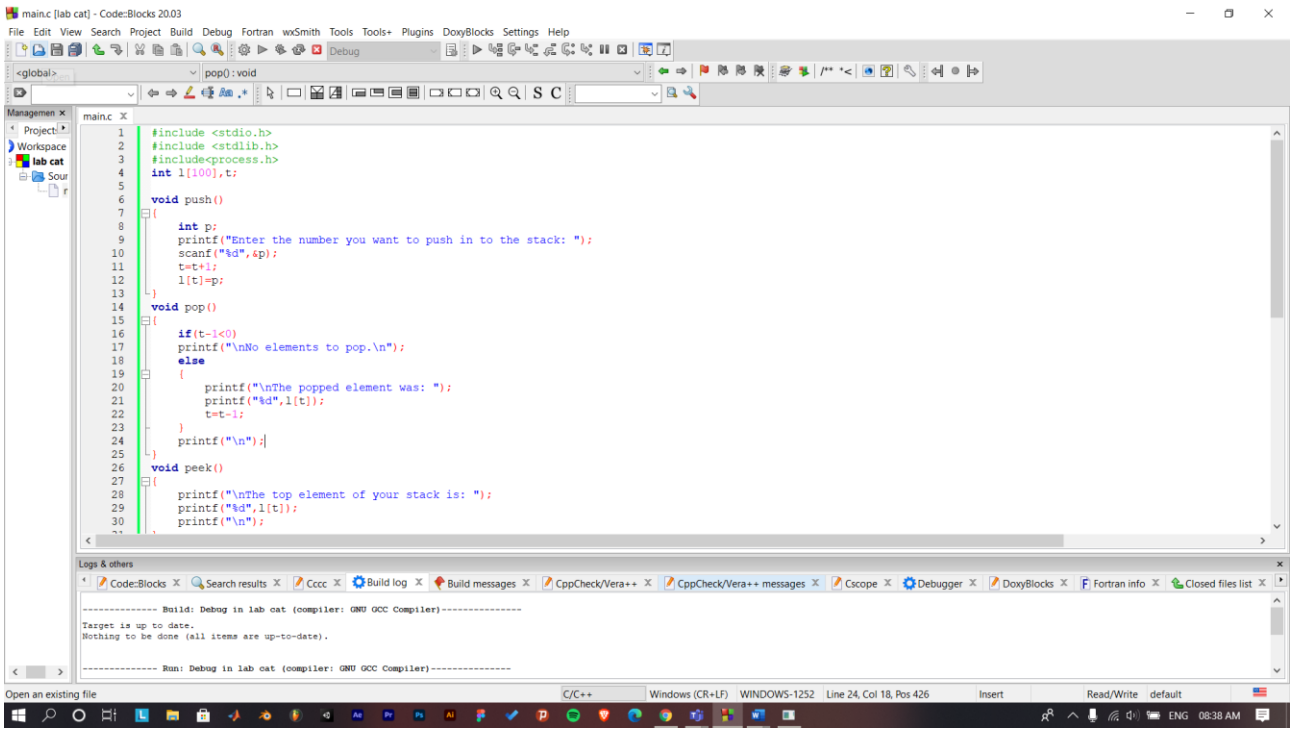


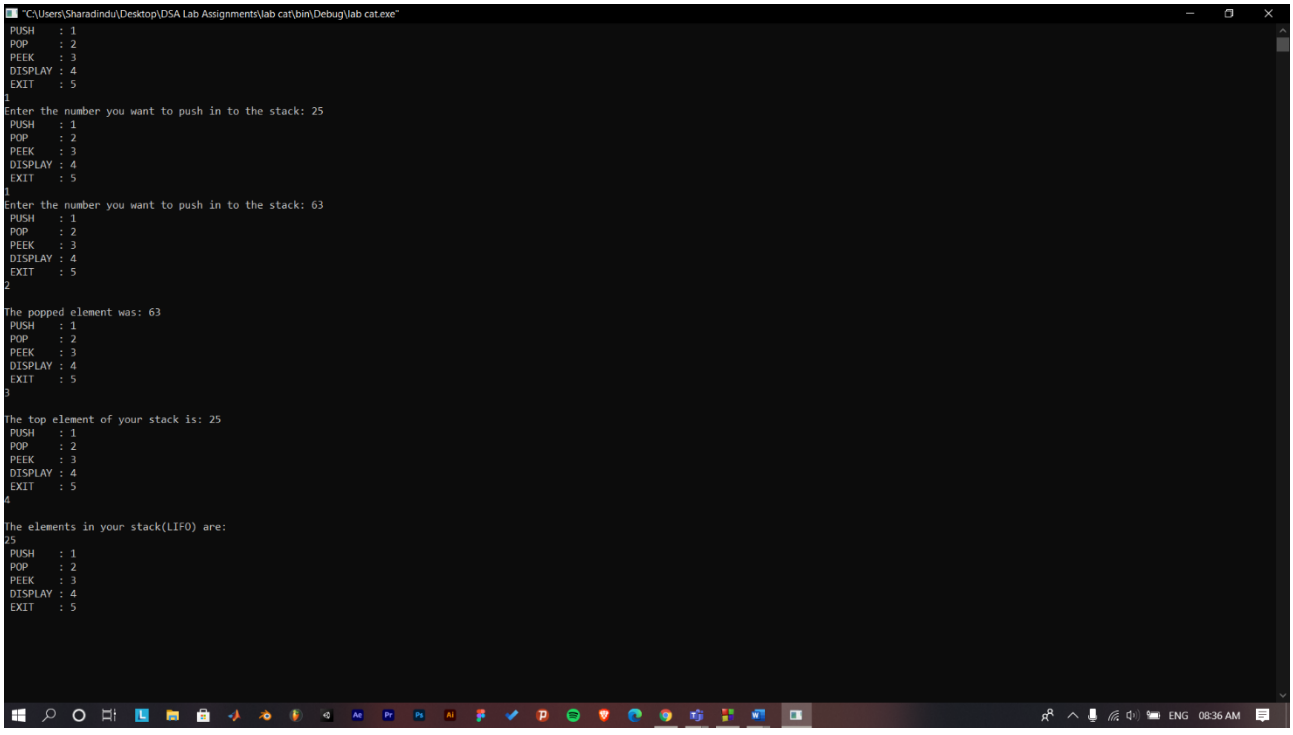


Using Array:

```
#include <stdio.h>
#include <stdlib.h>
#include<process.h>
int l[100],t;

void push()
{
    int p;
    printf("Enter the number you want to push in to the stack: ");
    scanf("%d",&p);
    t=t+1;
    l[t]=p;
}
void pop()
{
    if(t-1<0)
    printf("\nNo elements to pop.\n");
    else
    {
        printf("\nThe popped element was: ");
        printf("%d",l[t]);
        t=t-1;
    }
    printf("\n");
}
void peek()
{
    printf("\nThe top element of your stack is: ");
    printf("%d",l[t]);
    printf("\n");
}
void display()
{
    int i;
    printf ("\nThe elements in your stack(LIFO) are:\n");
    for (i=t;i>=0;i--)
    printf("%d ",l[i]);
    printf("\n");
}
int main()
{
    int R;
    t=-1;
    printf (" PUSH    : 1\n POP     : 2\n PEEK    : 3\n DISPLAY : 4\n EXIT    :
5\n");
    scanf("%d",&R);
    while(R!=5)
    {
        if(R==1)
        push();
        else if(R==2)
        pop();
        else if(R==3)
        peek();
        else if(R==4)
        display();
        printf (" PUSH    : 1\n POP     : 2\n PEEK    : 3\n DISPLAY : 4\n EXIT
: 5\n");
        scanf("%d",&R);
    }
    if (R==5)
```

```
    exit(0);
    return 0;
    getch();
}
```





Question 2

Problem:

Write a Program to convert an infix expression to prefix form.

Code, SS & CMD:

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>

#define BLANK ' '
#define TAB '\t'
#define MAX 50

long int pop();
long int eval_pre();
char infix[MAX], prefix[MAX];
long int stack[MAX];
int top;
int isempty();
int white_space(char symbol);

void infix_to_prefix();
int priority(char symbol);
void push(long int symbol);
long int pop();
long int eval_pre();

int main()
{
        long int value;
        top = -1;
        printf("Enter infix : ");
        gets(infix);
        infix_to_prefix();
        printf("prefix : %s\n",prefix);
        value=eval_pre();
        printf("Value of expression : %ld\n",value);

        return 0;

}/*End of main()*/

void infix_to_prefix()
{
        int i,j,p,n;
        char next ;
        char symbol;
        char temp;
        n=strlen(infix);
        p=0;

        for(i=n-1; i>=0; i--)
        {
                symbol=infix[i];
                if(!white_space(symbol))
                {
```

```
                        switch(symbol)
                        {
                        case ')':
                                push(symbol);
                                break;
                        case '(':
                                while( (next=pop()) != ')')
                                        prefix[p++] = next;
                                break;
                        case '+':
                        case '-':
                        case '*':
                        case '/':
                        case '%':
                        case '^':
                                while(  !isempty(  )  &&    priority(stack[top])>
priority(symbol) )
                                        prefix[p++] = pop();
                                push(symbol);
                                break;
                        default: /*if an operand comes*/
                             prefix[p++] = symbol;
                        }
                }
        }
        while(!isempty( ))
                prefix[p++] = pop();
        prefix[p] = '\0'; /*End prefix with'\0' to make it a string*/

        for(i=0,j=p-1;i<j;i++,j--)
        {
                temp=prefix[i];
                prefix[i]=prefix[j];
                prefix[j]=temp;
        }
}/*End of infix_to_prefix()*/

/* This function returns the priority of the operator */
int priority(char symbol )
{
        switch(symbol)
        {
        case ')':
                return 0;
        case '+':
        case '-':
                return 1;
        case '*':
        case '/':
        case '%':
                return 2;
        case '^':
                return 3;
        default :
                 return 0;
        }/*End of switch*/
}/*End of priority()*/

void push(long int symbol)
{
        if(top > MAX)
        {
                printf("Stack overflow\n");
                exit(1);
```

```
        }
        else
        {
                top=top+1;
                stack[top] = symbol;
        }
}/*End of push()*/

long int pop()
{
        if(top == -1 )
        {
                printf("Stack underflow \n");
                exit(2);
        }
        return (stack[top--]);
}/*End of pop()*/
int isempty()
{
        if(top==-1)
                return 1;
        else
                return 0;
}

int white_space(char symbol)
{
        if(symbol==BLANK || symbol==TAB || symbol=='\0')
                return 1;
        else
                return 0;
}/*End of white_space()*/

long int eval_pre()
{
        long int a,b,temp,result;
        int i;

        for(i=strlen(prefix)-1;i>=0;i--)
        {
                if(prefix[i]<='9' && prefix[i]>='0')
                        push( prefix[i]-48 );
                else
                {
                        b=pop();
                        a=pop();
                        switch(prefix[i])
                        {
                        case '+':
                                temp=b+a; break;
                        case '-':
                                temp=b-a;break;
                        case '*':
                                temp=b*a;break;
                        case '/':
                                temp=b/a;break;
                        case '%':
                                temp=b%a;break;
                        case '^':
                                temp=pow(b,a);
                        }
                        push(temp);
                }
        }
```

```
        result=pop();
        return result;
}/*End of eval_pre */
```