AIM

Design and implement of stack and demonstrate the working with necessary inputs and display the appropriate output.

ALGORTHIM:

1.PUSH:

Step 1:start.

Step 2:Create a new a node with the given data.

Step 3:check the wheather the stack is empty.

Step 4:If it is empty then set the pointer of the node to full.

Step 5:Else,then make the node point to head.

Step 6:make the new node created as head.

Step 7:stop.

2.POP:

Step 1:start

Step 2:Check wheather is empty.

Step 3:If empty,then display”stack is empty”.

Step 4:Enter create a temporary node and set it to the head.

Step 5:Print the data of head.

Step 6:make head to point the next node.

Step 7:Delete the temporary node.

Step 8:stop.

DISPLAY:

Step 1:start

Step 2:Create the temporary node and initialize it with the head pointer.

Step 3:Check if stack is empty,display”stack is empty”.

Step 4:Else,transverse the temporary node.If null is encountered.

PROGRAM

//implementation of stack using linked list//

#include<stdio.h>//standard input output header file//

#include<stdlib.h>//standard library header file//

void push();//push function//

void pop();//pop function//

void display();**//display function//**

int main()//main function//

{

int choice; //declaring the choice//

while(1)

{

printf("operation performed by stack\n");//prints the statement//

printf("1.push\n");//prints the functions//

printf("2.pop\n");//prints the functions//

printf("3.display\n");//prints the functions//

printf("4.exit\n");//prints the functions//

printf("enter choice\n");//print the statement//

scanf("%d",&choice); //scans the input and places the value into the choice//

switch(choice) //scans the condition and enters into the case when it satisfies//

{

case 1:push();//prints the case 1 when the condition is 1//

break;

case 2:pop();//prints the case 2 when the condition is 2//

break;

case 3:display();//prints the case 3 when the condition is 3//

break;

case 4:exit(0); //come out of the loop//

default:printf("invalid choice\n");//prints the value//

}

}

}

struct node//node decleration//

{

int val;

struct node \*next;

};

struct node \*head;

void push()

{

int val; //declaring the x//

struct node \*ptr=(struct node \*)malloc(sizeof(struct node));//dynamic memory allocation of the malloc function//

if(ptr==NULL)//checks the conditions//

{

printf("not able to push the element\n");//print the statement when the if satisfies//

}

else

{

printf("enter the value\n");//prints the statement//

scanf("%d",&val);

}

if(head==NULL)//checks the condition//

{

ptr->val=val;//pointer value//

ptr->next=NULL;

head=ptr;

}

else

{

ptr->val=val;

ptr->next=head;

head=ptr;

}

printf("item pushed\n");

}

void pop()

{

int item;

struct node \*ptr;

if(head==NULL)

{

printf("underflow\n");

}

else

{

item=head->val;

ptr=head;

head=head->next;

free(ptr);

printf("item popped\n");

}

}

void display()

{

int i;//declaration of i//

struct node \*ptr;

ptr=head;

if(ptr==NULL)

{

printf("stack is empty\n");

}

else

{

printf("printing stack\n");

while(ptr!=NULL)

{

printf("%d\n",ptr->val);

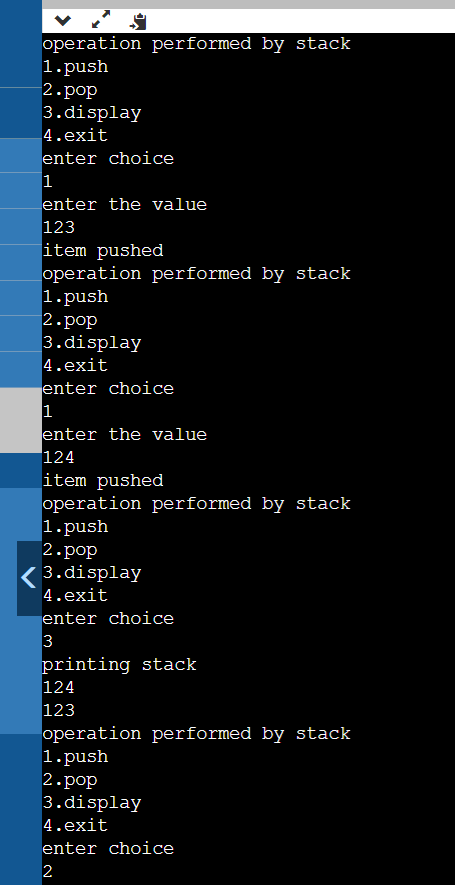
ptr=ptr->next;

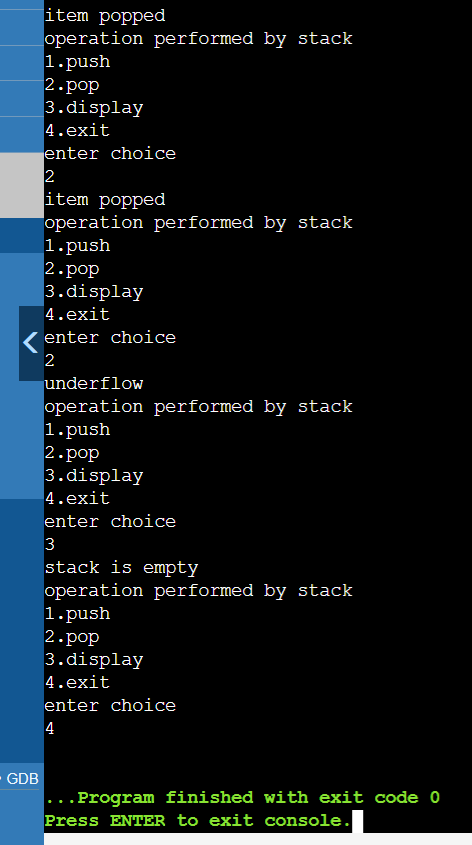
}

}

}

SCREENSHOT OF THE OUTPUT:





GITHUB LINK:

https://github.com/Rithwik314/DS-LAB-PROGRAMS/upload/main