**DS Lab**

**Operations on Stack**

**To Design and implement a stack (Array implementation/ Linked list implementation) and demonstrate its working with necessary inputs. Display the appropriate messages in case of exceptions.**

**Theory**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).

Basic Operations

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following two primary operations −

* push() − Pushing (storing) an element on the stack.
* pop() − Removing (accessing) an element from the stack.

When data is Pushed onto stack.

To use a stack efficiently, we need to check the status of stack as well. For the same purpose, the following functionality is added to stacks −

* peek() − get the top data element of the stack, without removing it.
* isFull() − check if stack is full.
* isEmpty() − check if stack is empty.

At all times, we maintain a pointer to the last Pushed data on the stack. As this pointer always represents the top of the stack, hence named top. The top pointer provides top value of the stack without actually removing it.

**Code**

#include <stdio.h> //includes standard input and output functions

#include <stdlib.h> //includes standard library functions

int s = 3; //stack size is declared to 3

int arr[3]; //a stack is created using array

int top=-1; //top is declared to -1 i.e empty stack

void push(int n) //function to push element in stack where n is the element

{

if(top>=2) //if top is greater than 2, the stack is //full as array starts from index 0 and //top is -1 for empty stack

printf("Stack is full\n"); //prints that stack //is full

else

{

top++; //top value gets incremented by 1

arr[top]=n; //the top index of array stores //the element

printf("%d was pushed in stack\n", arr[top]); //prints which element is pushed in stack

}

}

void pop() //function to pop element

{

int stackTop; //variable to store current topmost //element of stack

if(top<=-1) //if top value is -1, the stack is empty //and elements cannot be popped

printf("Stack is empty\n"); //print stack is //empty

else

{

stackTop=arr[top]; //stores topmost element of //stack in variable //stackTop

top--; //decrements top value by 1

printf("%d was popped from stack\n", stackTop); //print which element is popped out of stack

}

}

void stackTop() //function to display topmost element of stack

{

printf("%d is the topmost element of stack\n", arr[top]); //prints topmost element of stack

}

void display() //function to display stack in order

{

if(top>-1) //if stack is not empty then the code block gets executed

{

int i; //declare i variable for loop

printf("Stack:\n");

for(i=top; i>-1; i--) //for loop to print //stack in order from top to bottom

printf("%d\n", arr[i]); //prints the //element in current iteration

}

}

int main() //main function

{

int key, n; //declaration of key variable to select //options and n variable that stores value //to be pushed

while(1) //infinite while loop until certain condition //is met

{

printf("Enter: \n1 for Push\n2 for Pop\n3 for Stack Top\n4 for Stack Display\n5 to Quit\n"); //prints list //of options available

scanf("%d", &key); //stores key value

switch(key) //switch statement to choose //between operations

{

case 1: //case for push function

printf("Enter value to be pushed: "); //asks user for element to be pushed

scanf("%d", &n); //stores //element in n

push(n); //calls push function //with n as element

break; //break out of switch //statement

case 2: //case for pop function

pop(); //calls pop function

break; //break out of switch //statement

case 3: //case for stackTop function

stackTop(); //calls stackTop //function

break; //break out of switch //statement

case 4: //case for display function

display(); //calls display //function

break; //break out of switch //statement

case 5: //case to quit the program

exit(0); //function to //terminate the program

default: //case for when other options //are not selected

printf("Incorrect key\n"); //prints that the key is incorrect

break; //break out of switch //statement

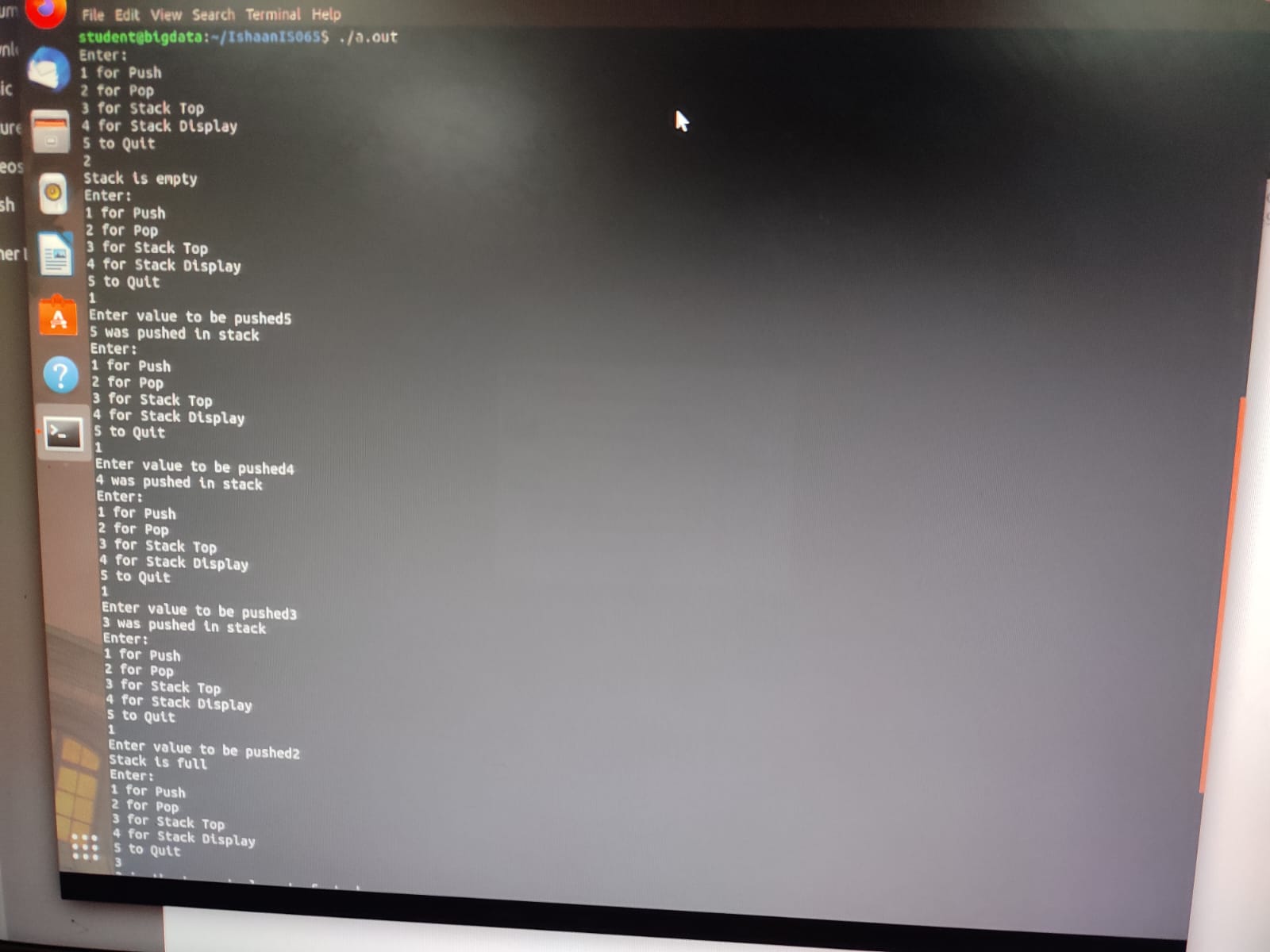
}

}

return 0; //return 0 tells that the program executes //without errors

}

**Output**

****