**Experiment :1**

**Title :** Implementation of Stack Linear Data Structure

**Problem Statement :** Implement linear Stack data structure which has functionalities of push(),pop(),display(),isempty(),ifull().

**Algorithm:**

**Step 1:**Start

**Step 2:** Declare stack array of type integer and a variable top =-1 to

keep the track of latest element added or removed

**Step 3:** Create function isfull() which checks for stack is full and if full return true else falls

**Step 4:** Create function isempty() which checks for stack is empty and if full return true else falls

**Step 5:** Create function push() which increments top variable and puts the value into the stack at index=top

**Step 6:** Create a function pop() function in which take the top element of stack and decrement top value

**Step 7:**Input user choice for push, pop, isfull, isempty or exit and switch according to the choice

**Step 8:** In switch call the function user has chosen and thus operate on stack

**Step 9:** Go to Step 7 till exit is chose

**Step 10:** Stop

**Program:**

**//**Stack

#include<stdio.h>

#include<stdlib.h>

# define MAX 5//max size of stack

int stack[MAX]; //Declare array stack of type int

int top = -1;

int i;

int isfull() //isfull function to check if stack is full

{

if(top == MAX-1)

{

printf("Stack is FULL\n");

return(1);

}

else

return(0);

}

int isempty() //isempty function to check for stack is empty

{

if(top == -1)

{

printf("Stack is EMPTY\n");

return(1);

}

else

return(0);

}

void push() //Declare push function

{

int value;

if(!isfull())//check if stack is full

{

printf("Enter the element to be pushed : ");

scanf("%d",&value);

top = top + 1;//increment top before pushing to move to next location

stack[top] = value;//push new value

printf("%d is pushed\n",stack[top]);

}

}

void pop() //Declare pop function

{

if(!isempty())//check if stack has any element or its empty

{

printf("%d is popped\n",stack[top]);//pop out the top element

top = top - 1;//decrement top to discard the popped element from stack

}

}

void display() //Display function

{

printf("\n\*\*\*\*STACK IS \*\*\*\*\n");

for(i=top;i>-1;--i)

{

printf("|\t%d\t|\n",stack[i]);

}

}

int main()

{

int choice;//to store the choice

printf("\*\*\*\* S T A C K D A T A S T R U C T U R E \*\*\*\*\*");

while(1)

{

printf("\nEnter your choice :\n1\_Push\t2\_Pop\t3\_Isempty\t4\_Isfull\t5\_Display Stack\t6\_Exit\n ");

scanf("%d",&choice); //input the operation

//switch the operation as per choice

switch(choice)

{

case 1 :push();

break;

case 2 :pop();

break;

case 3 :if(!isempty())

printf("\nStack is not empty");

break;

case 4 :if(!isfull())

printf("\nStack is not full");

break;

case 5 :display();

break;

case 6:exit(0);

break;

default :printf("Enter the correct choice\n");

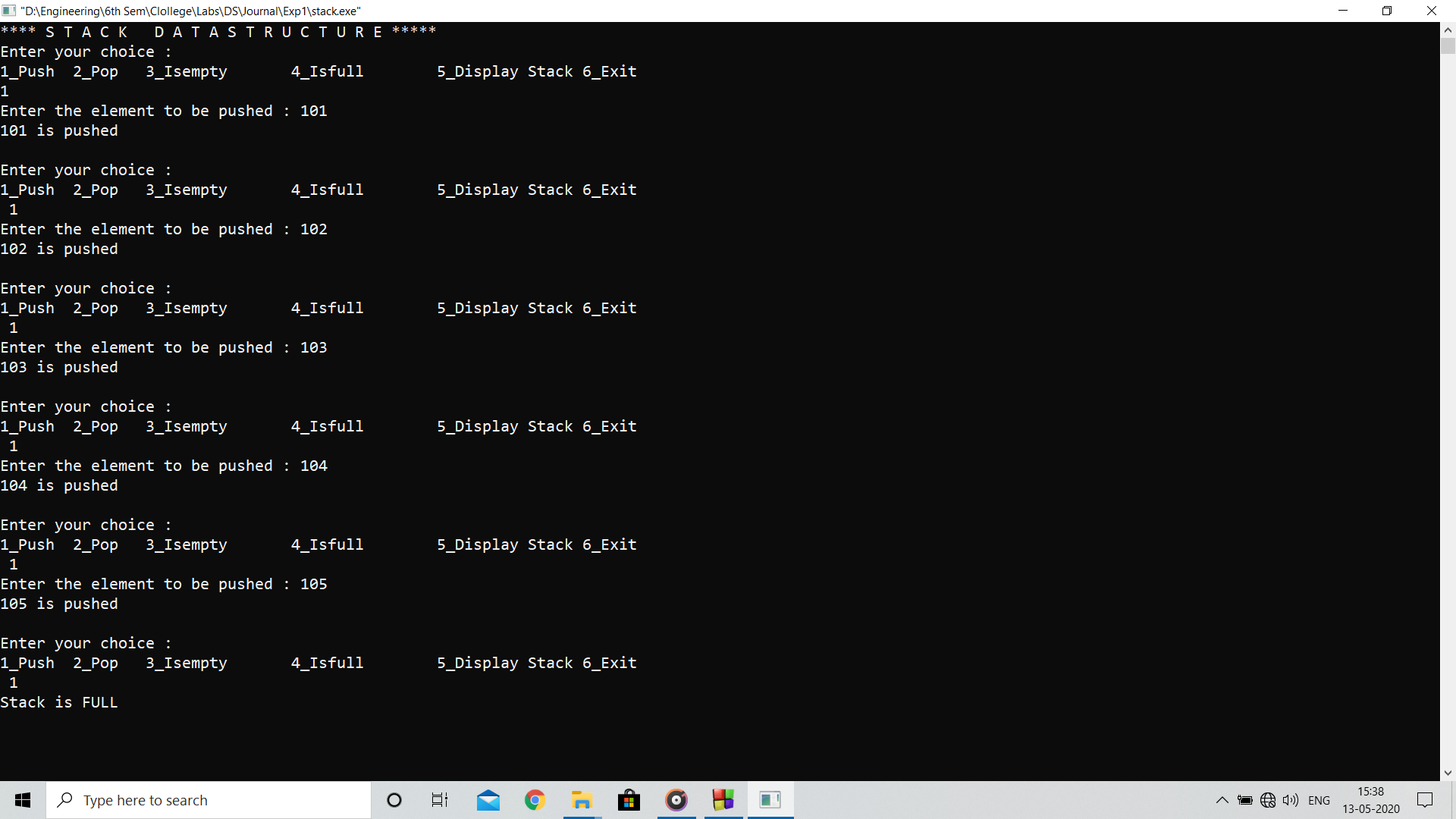
break;

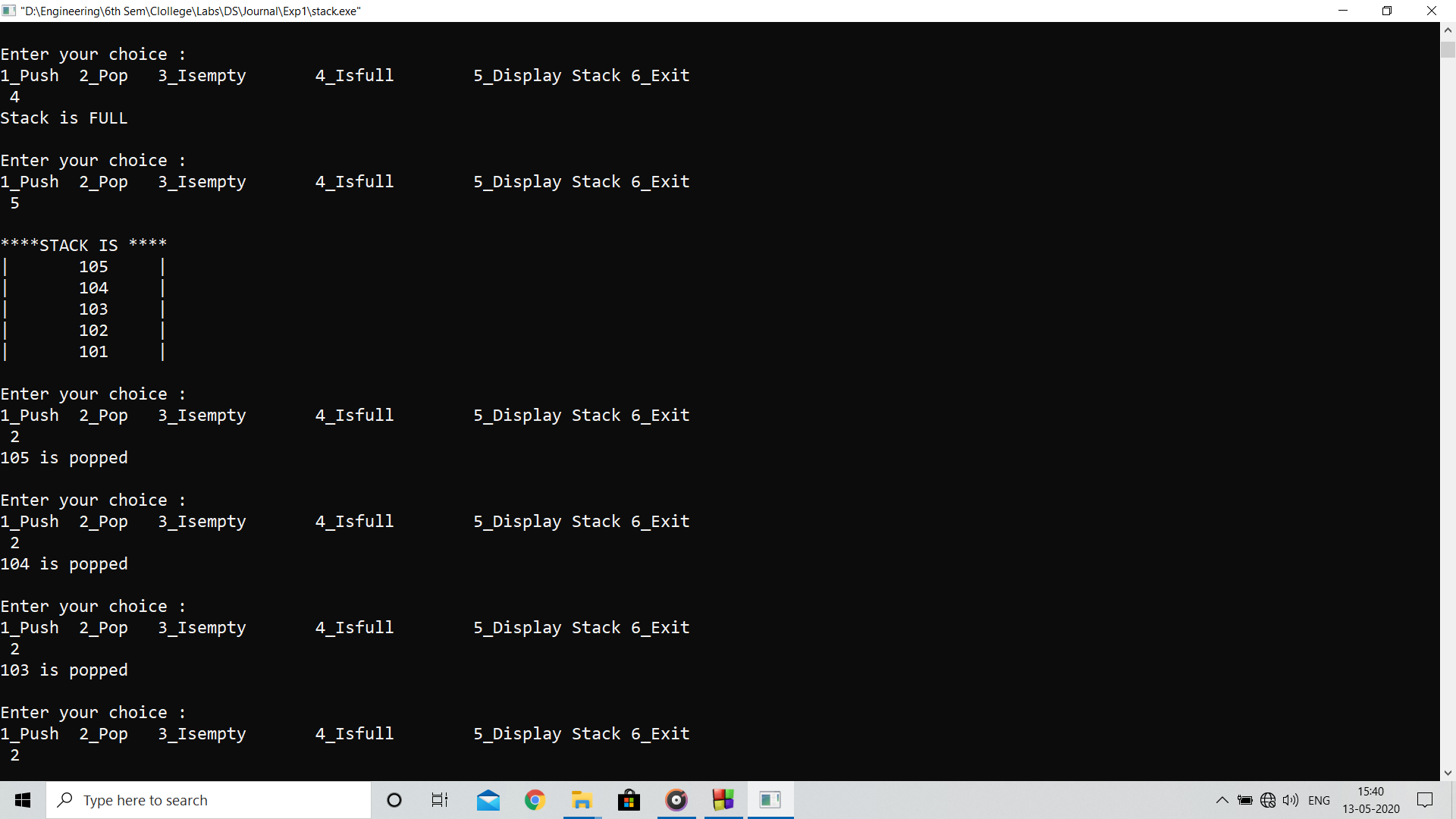
}//end switch

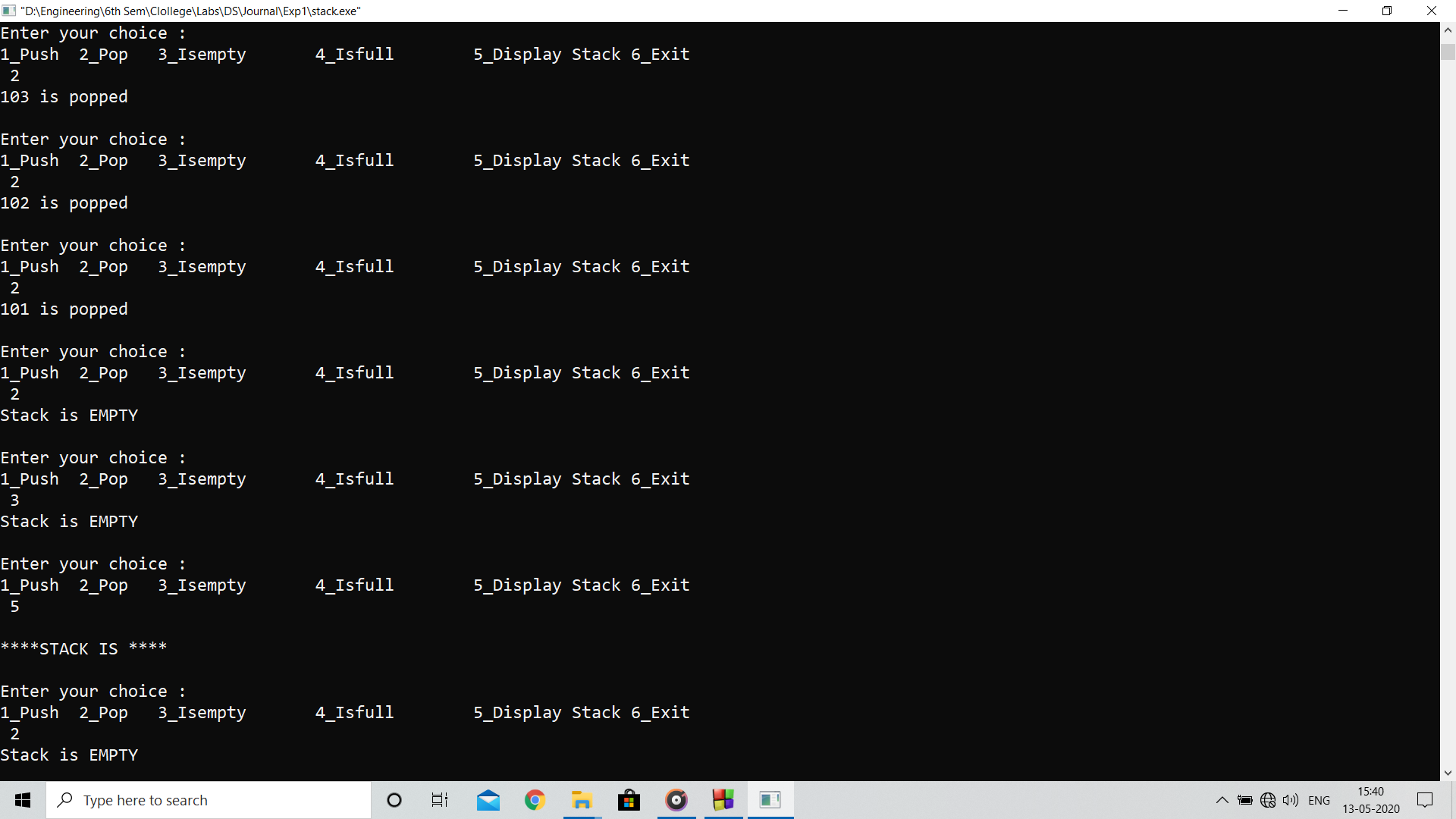
}//end while(1)

}//end main

**Output :**







**Analysis :**

Stack data structure is useful when only recent entries are important but if previous entries matter it fails to satisfy the operations considering time limits

**Limitations :**

Stack size is fixed in this program can be improved using dynamic memory allocation.