**Lab 8**

**Name:** Etcherla Sai Manoj **Mis. No:** 112015044 **Branch:** CSE

**Question1:**

**Code:**

#include<iostream>

using namespace std;

template<typename T>

void swap(T\* a,T\* b)

{

cout<<"Before swap : "<<\*a<<" and "<<\*b<<endl;

T t;

t = \*a;

\*a = \*b;

\*b = t;

cout<<"After swap : "<<\*a<<" and "<<\*b<<endl;

}

int main()

{

int choice;

cout << "===========TYPES===========\n";

cout<<"1.int\n2.double\n3.char\n";

cout << "==========================\n";

cout<<"Enter choice of your datatype : ";

cin>>choice;

switch(choice)

{

case 1:

{

int a,b;

cout<<"Enter two numbers you want to swap\n";

cout << "First one : ";

cin >> a;

cout << "Second one : ";

cin>>b;

swap<int>(&a,&b);

break;

}

case 2:

{

double a,b;

cout<<"Enter two double type you want to swap\n";

cout << "First one : ";

cin>>a;

cout << "Second one : ";

cin>>b;

swap<double>(&a,&b);

break;

}case 3:

{

char a,b;

cout<<"Enter 2 characters you want to swap\n";

cout << "First one : ";

cin>>a;

cout << "Second one : ";

cin>>b;

swap<char>(&a,&b);

break;

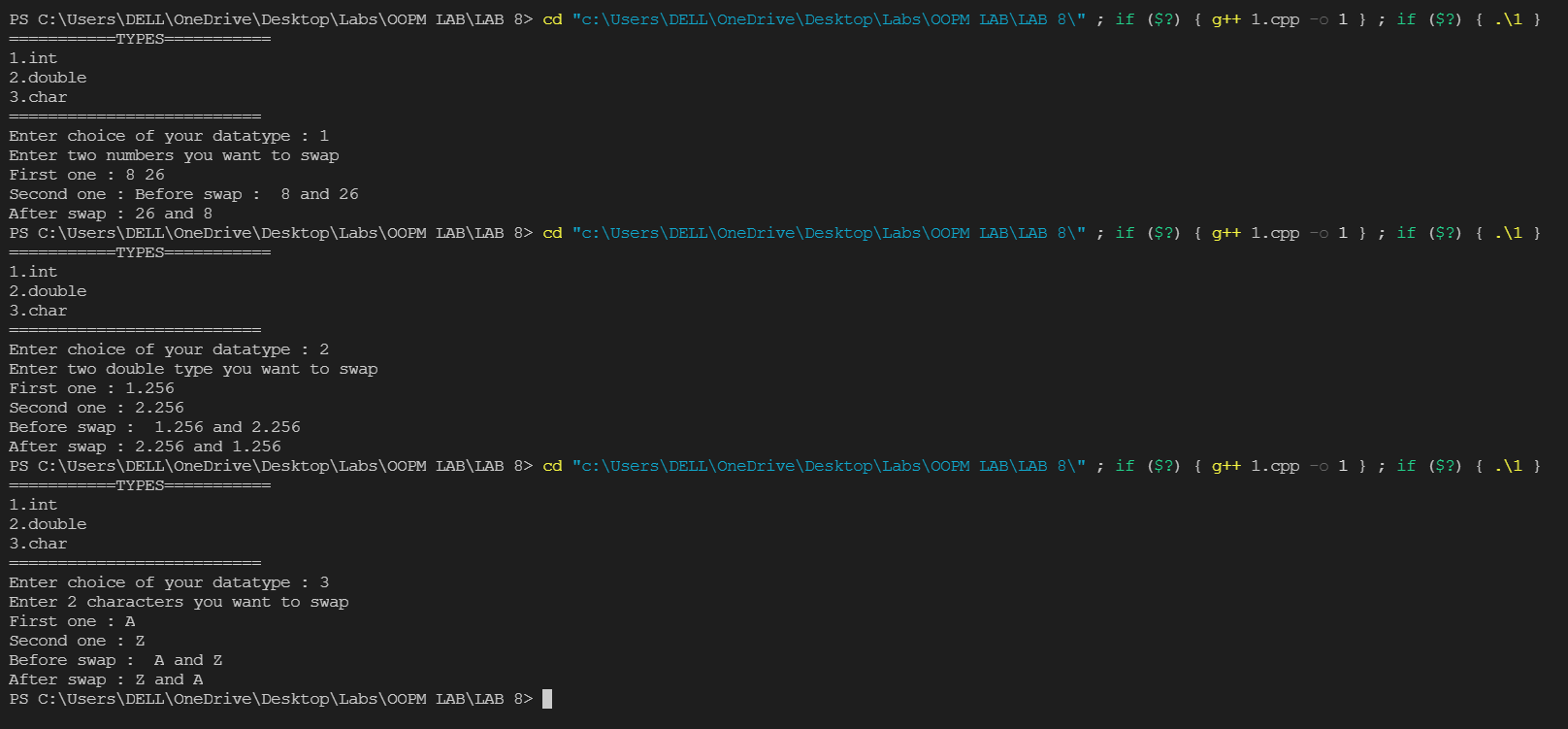
}

}

return 0;

}

**Input & output:**

****

**Question2:**

**Code:**

#include<bits/stdc++.h>

using namespace std;

template<class T>

void exception\_h()

{

string sentence;

cout<<"Enter a string : ";

cin>>sentence;

T i=0 ;

T last = sentence[0];

while(sentence[i]!='\0')

{

try{

if(sentence[i]=='!'|| sentence[i]=='#'||sentence[i]=='&'||sentence[i]==1||sentence[i]==2||sentence[i]==3||sentence[i]==4||sentence[i]==5||sentence[i]==6||sentence[i]==7||sentence[i]==8||sentence[i]==9)

{

throw sentence[i];

}

else {

if(sentence[i]>last)

{

last = sentence[i];

}

}

}

catch (...)

{

cout<<"Expection Caught!!! unwanted characted are used..\n";

}

i++;

}

cout<<"Alphabetically last character is : "<<char(last)<<endl;

}

int main()

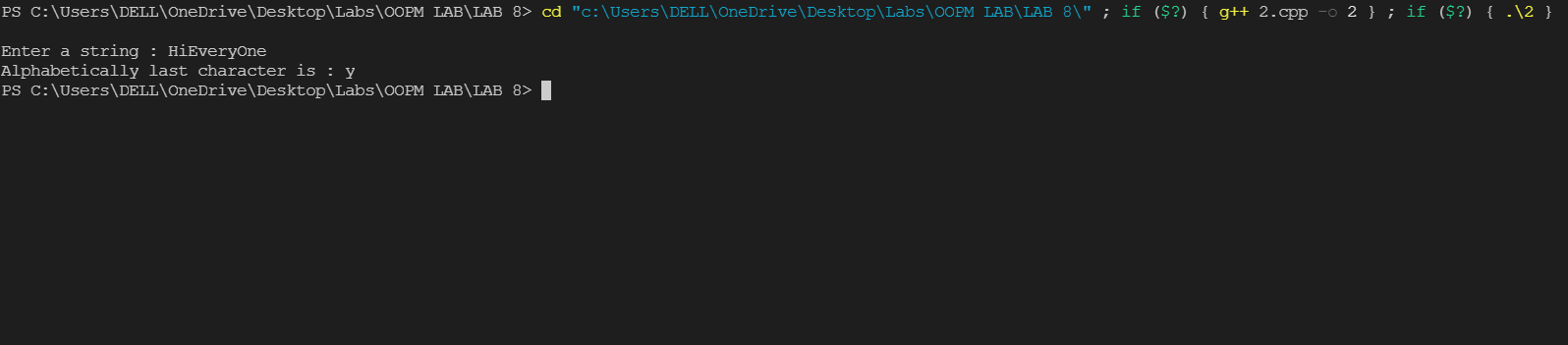
{

exception\_h<int>();

return 0;

}

**Input & output:**

****

**Question3:**

**Code:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

//division with zero

int a = 2;

int b = 2;

int c = 5;

try

{

if(a-b != 0){

float d = c / (a-b);

}

else{

throw (a-b);

}

}

catch (int x){

cout << "\nException caught for (a-b)" << endl;

}

//square root of negative number

int t = -2;

try{

if(t > 0){

float e = sqrt(t);

}

else{

throw t;

}

}

catch (int y){

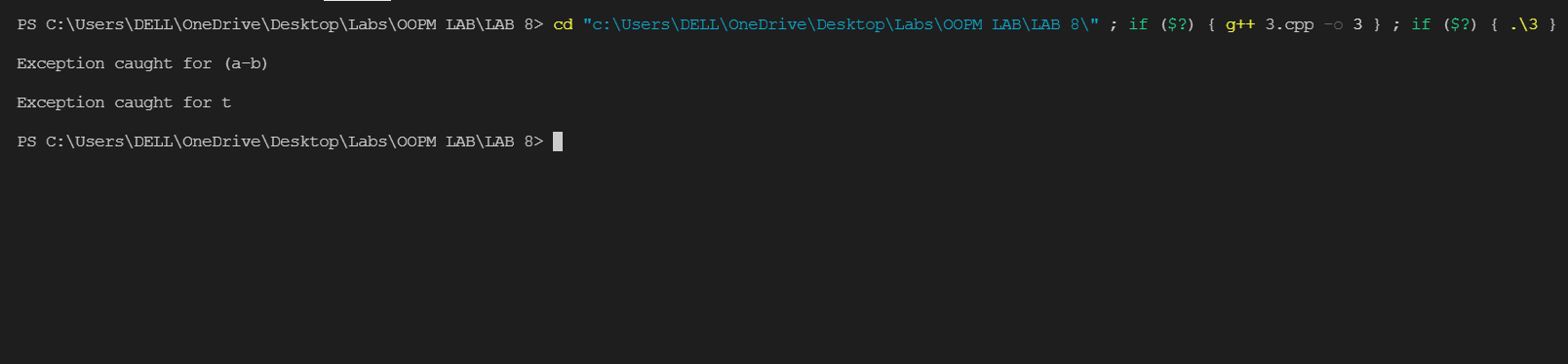
cout << "\nException caught for t\n" << endl;

}

return 0;

}

**Input & output:**

****

**Question4:**

**Code:**

#include<iostream>

using namespace std;

template<class T>

class Stack{

int top;

int top\_most;

T \*S;

public:

Stack(int max\_size);

~Stack(){

delete[] S;

}

int is\_empty()const{

return top==-1;

}

int is\_full()const{

return top==top\_most;

}

T peek()const;

void push(T);

T pop();

void display();

};

template<class T>

Stack<T>::Stack(int max\_size)

{

top\_most=max\_size-1;

S=new T[max\_size];

top=-1;

}

template<class T>

T Stack<T>::peek()const

{

if(is\_empty())

return 0;

else

return S[top];

}

template<class T>

void Stack<T>::push(T x)

{

if(is\_full())

cout<<"Stack is full\n";

else

{

S[++top]=x;

}

}

template<class T>

T Stack<T>::pop()

{

T x;

if(is\_empty())

{

cout<<"Stack is empty\n";

return -1;

}

else

{

x=S[top--];

return x;

}

}

template<class T>

void Stack<T>::display()

{

if(is\_empty())

cout<<"Out of bounds";

else

cout << "Elements of stack : ";

for(int i=top;i>=0;i--)

{

cout<<S[i]<<" ";

}

cout << "\n";

}

int main()

{

Stack<int>obj(5);

int ch,x;

cout << "===========MENU===========\n";

cout<<"1.push\n2.pop\n3.peek\n4.display\n5.Exit\n";

cout << "==========================\n";

while(1){

cout<<"\nEnter the choice : ";

cin>>ch;

switch(ch){

case 1:

cout<<"Enter a value to push into the stack : ";

cin>>x;

obj.push(x);

break;

case 2:

x=obj.pop();

if(x!=-1)

cout<<"Poped value is : "<<x<<endl;

break;

case 3:

x=obj.peek();

cout<<"Top most value is : "<<x<<endl;

break;

case 4:

obj.display();

break;

case 5:

return 0;

break;

default:

cout << "Enter valid choice...!\n";

break;

}

}

return 0;

}

**Input & output:**

