Q1)a)

CODE:-

#include <iostream>

#include <cmath>

using namespace std;

class Rectangle;

class Polar

{

float radius, angle;

public:

Polar()

{

radius = 0;

angle = 0;

}

void getPolar()

{

cout << "Enter Radius and Angle.\n";

cin >> radius >> angle;

}

Polar(Rectangle &);

void display()

{

cout << "The Radius = " << radius << endl;

cout << "The Angle = " << angle << endl;

}

float retradius()

{

return radius;

}

float retangle()

{

return angle;

}

};

class Rectangle

{

float x, y;

public:

Rectangle()

{

x = 0;

y = 0;

}

void getRectangle()

{

cout << "Enter X and Y coordinates.\n";

cin >> x >> y;

}

Rectangle(Polar &p)

{

x = p.retradius() \* cos(p.retangle());

y = p.retradius() \* sin(p.retangle());

}

void display()

{

cout << "The X coordinate is " << x << "\n"

<< "The Y coordinate is " << y << endl;

}

float retx()

{

return x;

}

float rety()

{

return y;

}

};

Polar::Polar(Rectangle &r)

{

float a;

a = atan((r.rety() / r.retx()));

angle = a \* (180 / 3.14);

radius = sqrt((r.retx() \* r.retx() + r.rety() \* r.rety()));

}

int main()

{

Polar p1;

Rectangle r1;

int ch;

while (1)

{

cout << "1.Polar to Rectangle.\n";

cout << "2.Rectangle to polar.\n";

cout << "3.Exit.\n";

cout << "Enter your choice.\n";

cin >> ch;

switch (ch)

{

case 1:

p1.getPolar();

r1 = p1;

r1.display();

break;

case 2:

r1.getRectangle();

p1 = r1;

p1.display();

break;

case 3:

exit(1);

default:

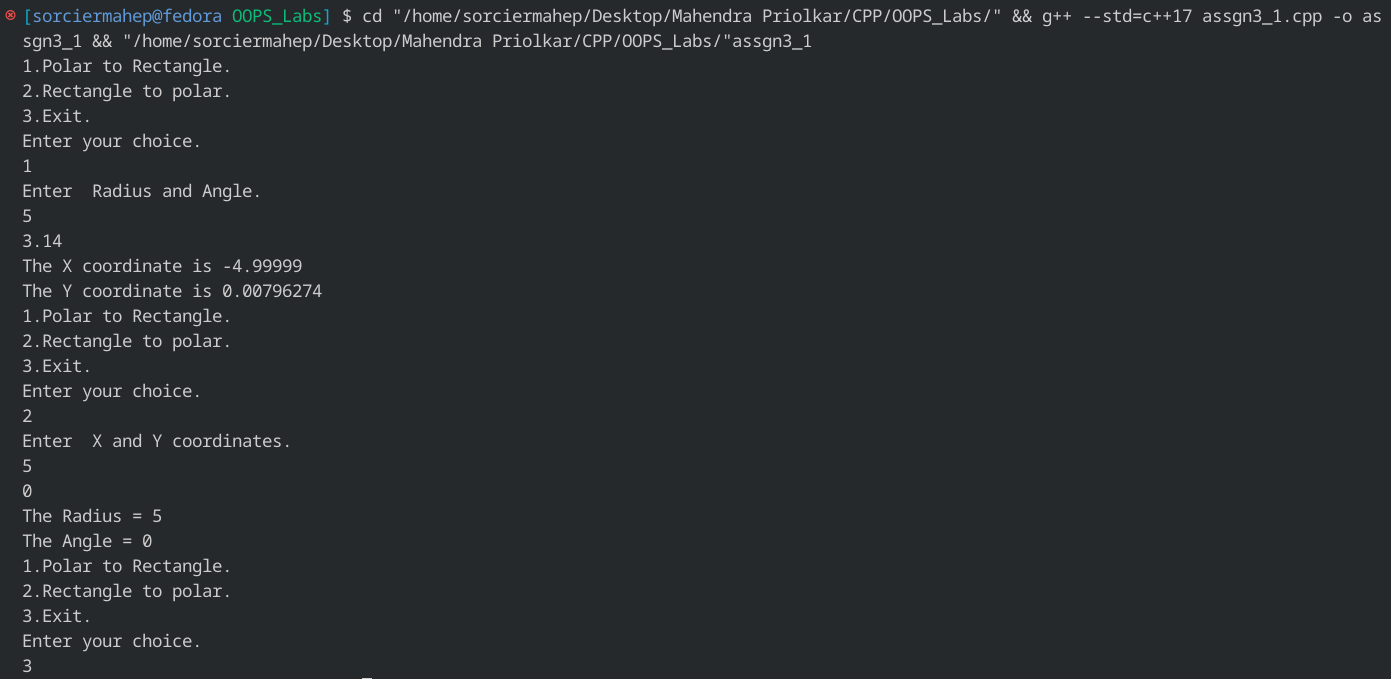
cout << "Erroneous input.\n";

}

}

}

OUTPUT:-



Q1)b)

CODE:-

#include <iostream>

using namespace std;

class list

{

char name[20];

int code;

float price;

public:

void getinfo(int i)

{

cout << "Enter name of item " << i << "\n";

getchar();

cin.getline(name, 20);

cout << "Enter code of item " << i << " \n";

cin >> code;

cout << "Enter price of item " << i << " \n";

cin >> price;

}

void display()

{

cout.setf(ios::showpoint);

cout.precision(5);

cout.setf(ios::left, ios::adjustfield);

cout.width(10);

cout << name;

cout.width(7);

cout << code;

cout.width(8);

cout.setf(ios::right, ios::adjustfield);

cout << price;

cout << "\n";

}

};

int main()

{

list \*l;

int n;

cout << "Enter total no. of list items.\n";

cin >> n;

l = new list[n];

for (int i = 0; i < n; i++)

l[i].getinfo(i + 1);

cout.fill('-');

cout.width(25);

cout << "\n";

cout.setf(ios::left, ios::adjustfield);

cout.fill(' ');

cout.width(10);

cout << "Name";

cout.width(7);

cout << "Code";

cout.width(8);

cout << "Price";

cout.fill('-');

cout.width(25);

cout << "\n";

cout.fill(' ');

cout << "\n";

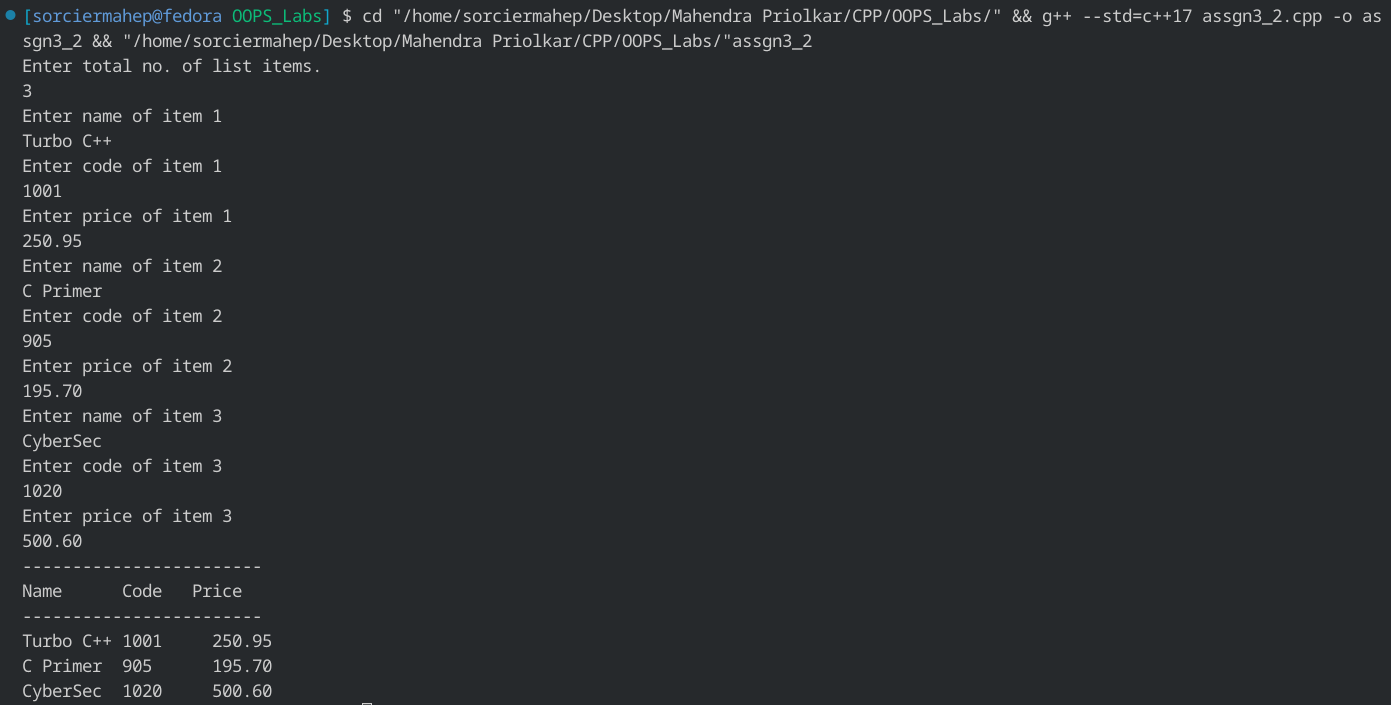
for (int i = 0; i < n; i++)

l[i].display();

return 0;

}

OUTPUT:-



Q2)a)

CODE:-

#include <iostream>

#include <cstring>

using namespace std;

class str

{

char \*s;

int len;

public:

str()

{

len = 0;

s = new char[len + 1];

strcpy(s, "");

}

str(char \*str)

{

len = strlen(str);

s = new char[len + 1];

s = strcpy(s, str);

}

friend str operator+(str &, str &);

void operator=(str);

void display()

{

cout << s << endl;

}

~str()

{

cout << "Destructor called." << endl;

}

};

str operator+(str &n, str &m)

{

str temp;

temp.len = strlen(n.s) + strlen(m.s);

temp.s = new char[temp.len + 1];

strcpy(temp.s, n.s);

strcat(temp.s, m.s);

return temp;

}

void str::operator=(str strn)

{

strcpy(s, strn.s);

}

int main()

{

str s1("Sorcier ");

str s2("MaheP");

str s3;

s3 = s1 + s2;

cout << "FIRST STRING : ";

s1.display();

cout << "SECOND STRING : ";

s2.display();

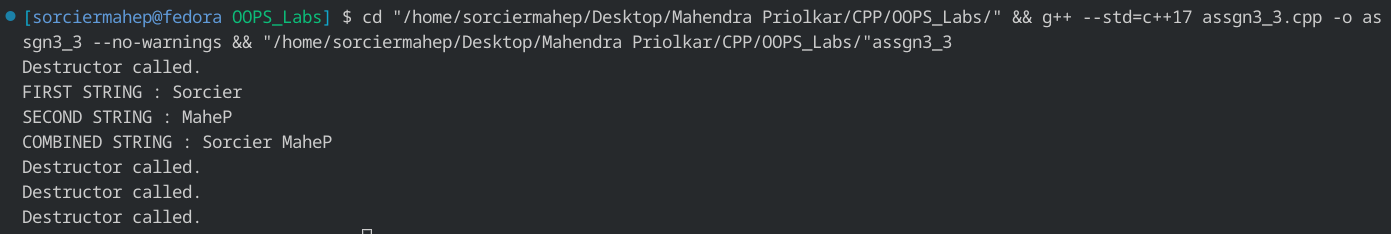
cout << "COMBINED STRING : ";

s3.display();

return 0;

}

OUTPUT:-



Q2)b)

CODE:-

# include <iostream>

using namespace std;

class Base

{

public:

virtual void display()

{

cout<<"Display base class."<<endl;

}

virtual void show()

{

cout<<"Show base class."<<endl;

}

};

class Derived:public Base

{

public:

void display()

{

cout<<"Display derived class."<<endl;

}

void show()

{

cout<<"Show derived class."<<endl;

}

};

int main()

{

Base B;

Derived D;

Base \*bptr;

bptr=&B;

cout<<"bptr points to base."<<endl;

bptr->display();

bptr->show();

bptr=&D;

cout<<"bptr points to derived."<<endl;

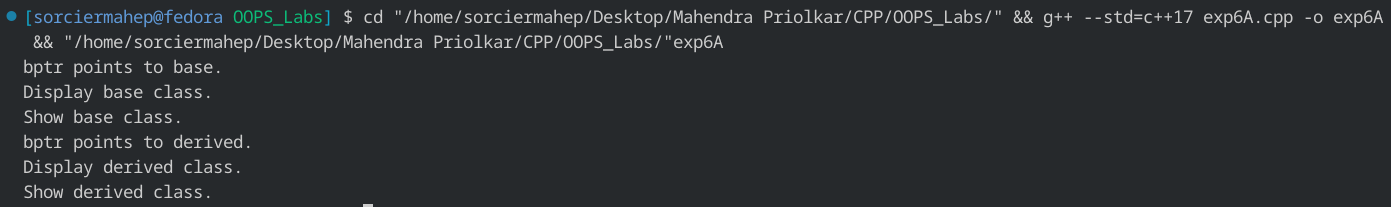
bptr->display();

bptr->show();

return 0;

}

OUTPUT:-



Q3)a)

CODE:-

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

int main()

{

char arr[1000], c;

int i = 0;

fstream f1;

f1.open("input.txt", ios::in);

f1.get(c);

while (f1.good())

{

arr[i++] = c;

f1.get(c);

}

arr[i] = '\0';

cout << "Contents of the file are:\n"

<< arr << endl;

f1.close();

f1.open("output.txt", ios::out);

for (int j = 0; j < i; j++)

{

c = arr[j];

if (c == ' ')

{

while (arr[j + 1] == ' ' && j < i)

j++;

}

f1.put(c);

}

f1.close();

f1.open("output.txt");

cout << "The resultant file is:\n";

f1.get(c);

while (f1.good())

{

cout << c;

f1.get(c);

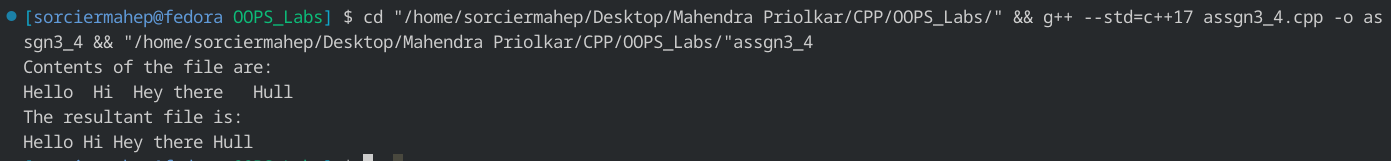
}

f1.close();

cout << endl;

return 0;

}

OUTPUT:-

Q3)b)

CODE:-

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

class telephone

{

char name[20];

int phone;

public:

char \*ret\_name()

{

return name;

}

int ret\_phone()

{

return phone;

}

void getname(char \*str)

{

strcpy(name, str);

}

void getphone(int num)

{

phone = num;

}

};

void display\_phone(char \*str, int i, telephone t[])

{

int flag = 0;

for (int j = 0; j < i; j++)

{

if (strcmp(str, t[j].ret\_name()) == 0)

{

cout << "Phone number is: " << t[j].ret\_phone() << endl;

flag = 1;

break;

}

}

if (flag == 0)

cout << "Record does not exist." << endl;

}

void display\_name(int num, int i, telephone t[])

{

int flag = 0;

for (int j = 0; j < i; j++)

{

if (num == t[j].ret\_phone())

{

cout << "Name is: " << t[j].ret\_name() << endl;

flag = 1;

break;

}

}

if (flag == 0)

cout << "Record does not exist." << endl;

}

void modify\_telephone(char \*str, int i, telephone t[])

{

int flag = 0, newnum;

for (int j = 0; j < i; j++)

{

if (strcmp(str, t[j].ret\_name()) == 0)

{

cout << "Enter the new phone number." << endl;

cin >> newnum;

t[j].getphone(newnum);

fstream f1;

f1.open("temp.txt", ios::out);

char str[20];

int num;

for (int j = 0; j < i; j++)

{

f1 << t[j].ret\_name();

f1.put(' ');

f1 << t[j].ret\_phone();

f1.put('\n');

}

f1.close();

remove("tel.txt");

rename("temp.txt", "tel.txt");

flag = 1;

break;

}

}

if (flag == 0)

cout << "Record does not exist." << endl;

}

int main()

{

int ch, i = 0, tempnum, num;

char str[20], tempstr[20];

telephone t[10];

fstream f;

f.open("tel.txt", ios::in);

f >> tempstr;

while (f.good())

{

t[i].getname(tempstr);

f >> tempnum;

t[i++].getphone(tempnum);

f >> tempstr;

}

while (1)

{

cout << "1.Search a Telephone Record as per name." << endl;

cout << "2.Search a name as per Telephone Record." << endl;

cout << "3.Modify a Telephone Record as per name." << endl;

cout << "4.Exit." << endl;

cin >> ch;

switch (ch)

{

case 1:

cout << "Enter the name." << endl;

cin >> str;

display\_phone(str, i, t);

break;

case 2:

cout << "Enter the number." << endl;

cin >> num;

display\_name(num, i, t);

break;

case 3:

cout << "Enter the name." << endl;

cin >> str;

modify\_telephone(str, i, t);

break;

case 4:

exit(1);

default:

cout << "Erroneous input." << endl;

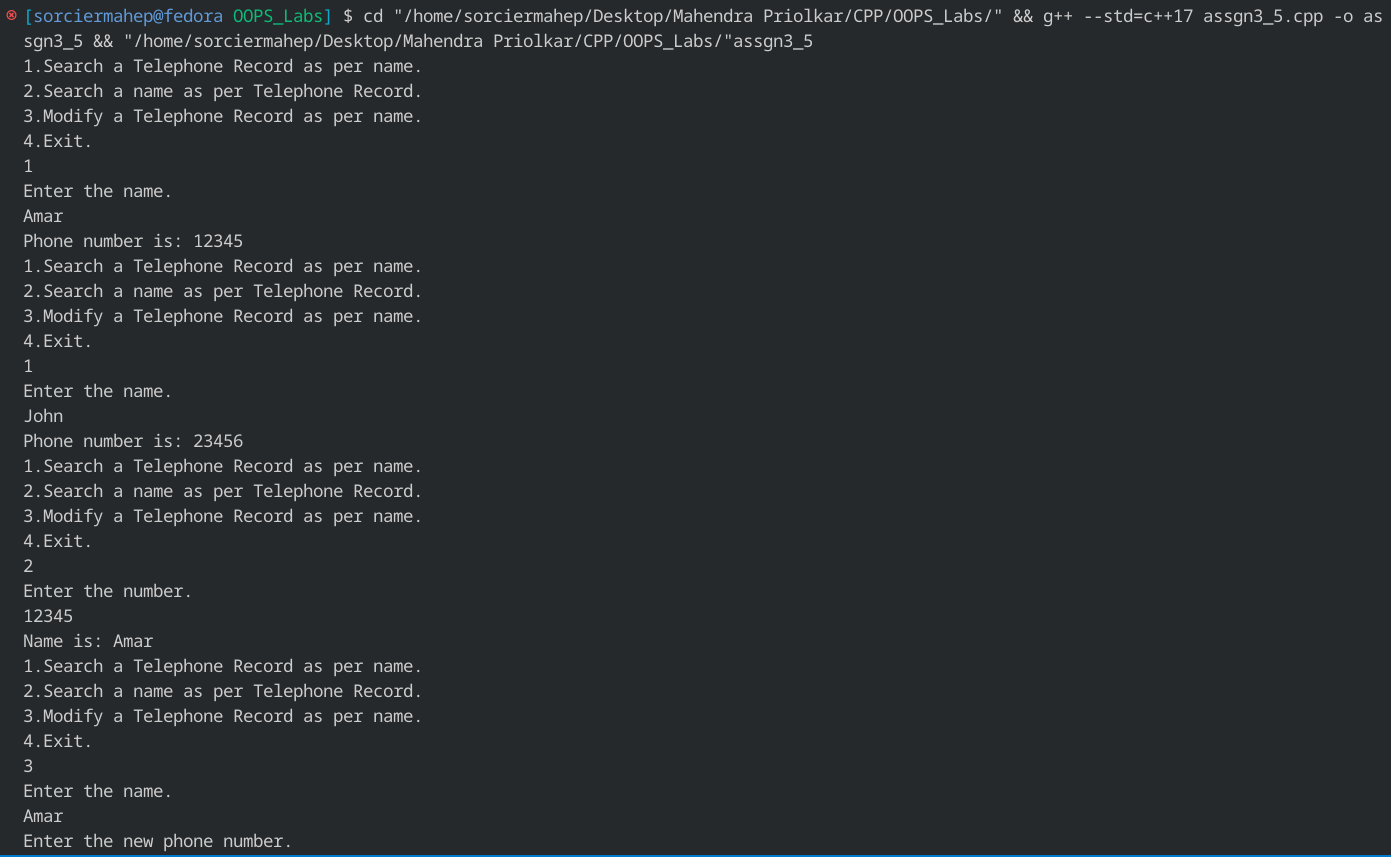
}

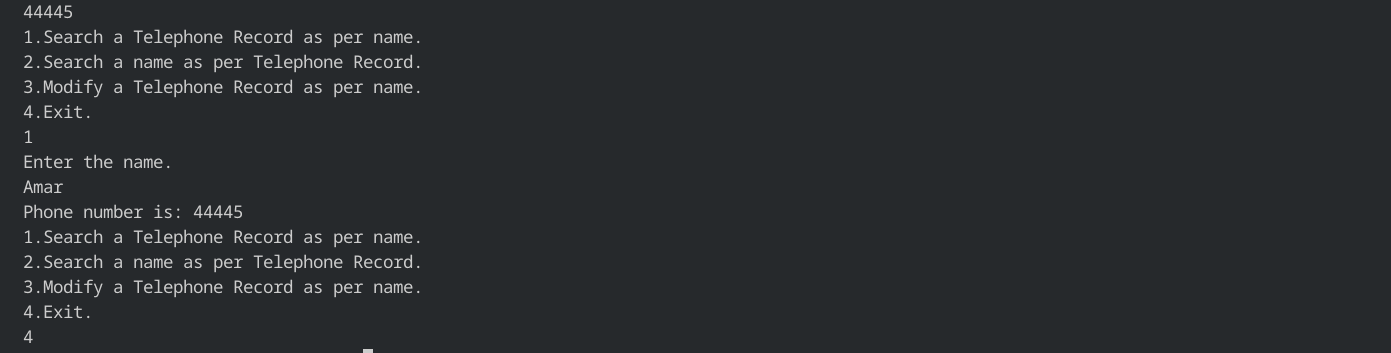
}

return 0;

}

OUTPUT:-





Q4)a)

CODE:-

#include <iostream>

using namespace std;

int main()

{

double a;

try

{

cout << "Enter value." << endl;

cin >> a;

if (a > 9999)

throw a;

else

cout << "Input= " << a;

}

catch (double a)

{

cout << "Error caught.Input greater than 9999." << endl;

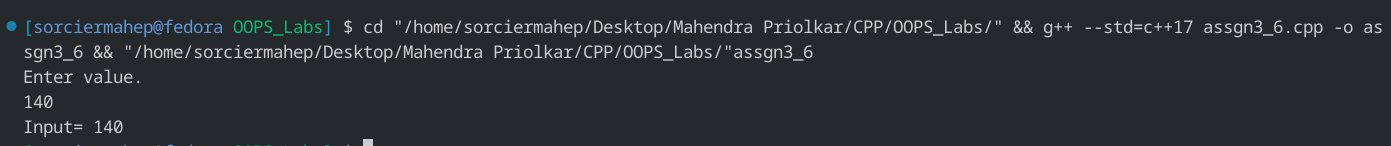
}

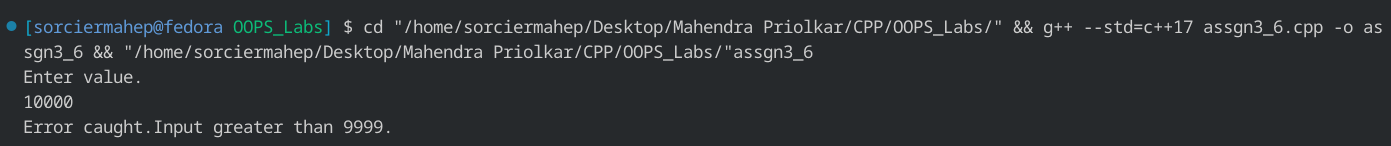
cout << endl;

return 0;

}

OUTPUT:-





Q4)b)

CODE:-

#include <iostream>

using namespace std;

template <class T>

void exchange(T &a, T &b)

{

T temp;

temp = a;

a = b;

b = temp;

}

int main()

{

int i1 = 5, i2 = 10;

cout << "For int." << endl;

cout << "i1= " << i1 << " i2= " << i2 << endl;

exchange(i1, i2);

cout << "i1= " << i1 << " i2= " << i2 << endl;

float f1 = 5.5, f2 = 10.56;

cout << "For float." << endl;

cout << "f1= " << f1 << " f2= " << f2 << endl;

exchange(f1, f2);

cout << "f1= " << f1 << " f2= " << f2 << endl;

char c1 = 'a', c2 = 'b';

cout << "For char." << endl;

cout << "c1= " << c1 << " c2= " << c2 << endl;

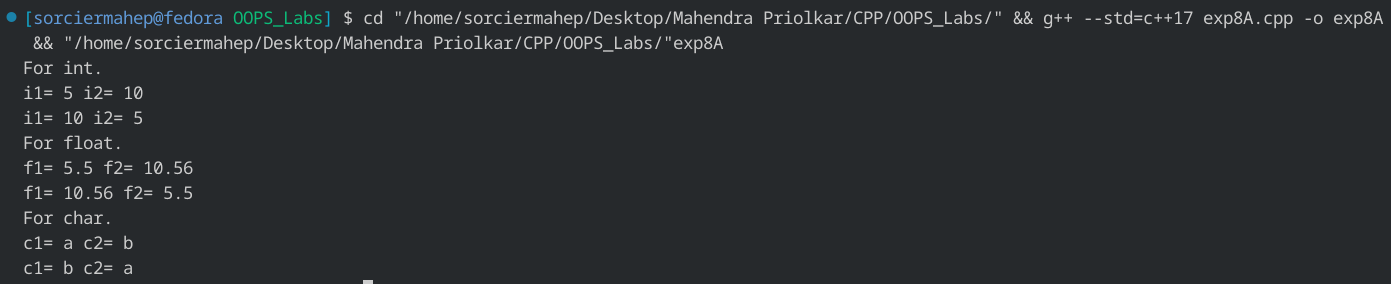
exchange(c1, c2);

cout << "c1= " << c1 << " c2= " << c2 << endl;

return 0;

}

OUTPUT:-



Q4)c)

CODE:-

#include <iostream>

using namespace std;

void func1()

{

cout << "Function 1 invoked.\n";

throw 1;

cout << "Function 1 completed.\n";

}

void func2()

{

cout << "Function 2 invoked.\n";

func1();

cout << "Function 2 completed.\n";

}

void func3()

{

cout << "Function 3 invoked.\n";

try

{

func2();

}

catch (int a)

{

cout << "Error Caught.\n";

}

cout << "Function 3 Completed.\n";

}

int main()

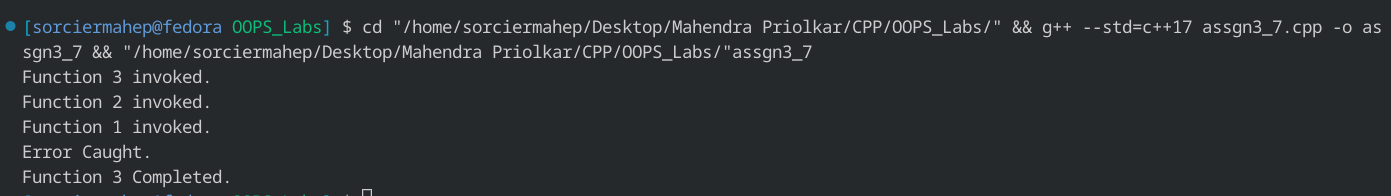
{

func3();

return 0;

}

OUTPUT:-



Q5)a)

CODE:-

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main()

{

string s1, s2;

cout << "Enter The string." << endl;

cin >> s1;

s2 = s1;

reverse(s1.begin(), s1.end());

if (s2 == s1)

cout << "The string is a palindrome." << endl;

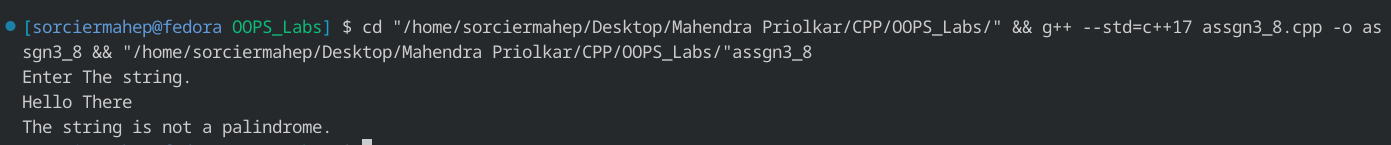
else

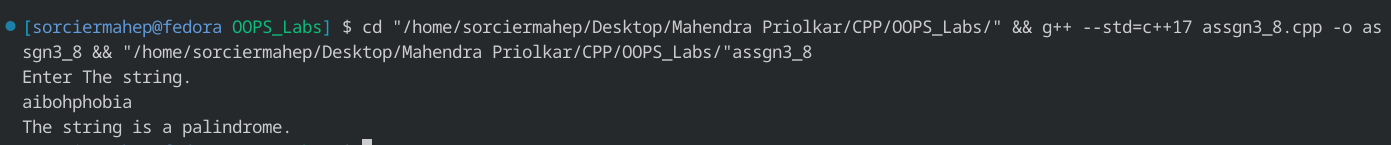
cout << "The string is not a palindrome." << endl;

return 0;

}

OUTPUT:-





Q5)b)

CODE:-

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main()

{

string s1;

cout << "Enter the string." << endl;

cin >> s1;

cout << "substr(0,3) function." << endl

<< s1.substr(0, 3) << endl;

cout << "capacity() function." << endl

<< s1.capacity() << endl;

cout << "find('h') function." << endl

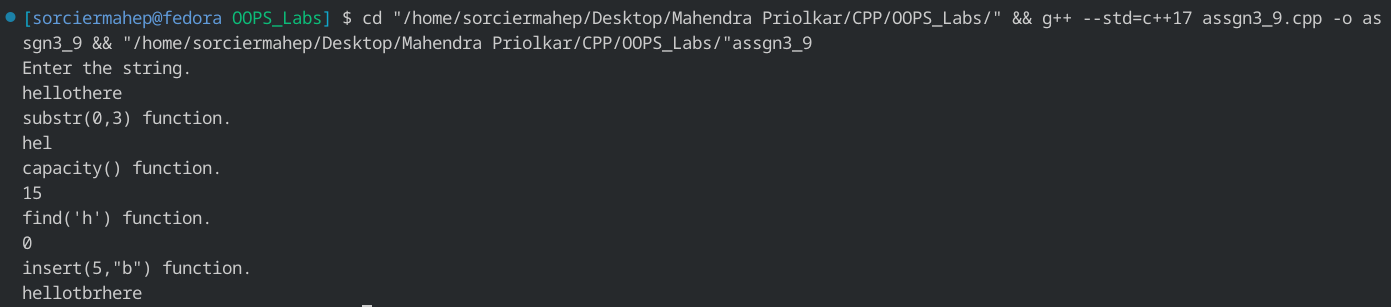
<< s1.find('h') << endl;

cout << "insert(5,\"b\") function." << endl

<< s1.insert(6, "br") << endl;

}

OUTPUT:-



Q5)c)

CODE:-

#include <iostream>

#include <string>

using namespace std;

int main()

{

string s1;

cout << "Enter the string." << endl;

cin >> s1;

for (string::iterator itr = s1.begin(); itr != s1.end(); itr++)

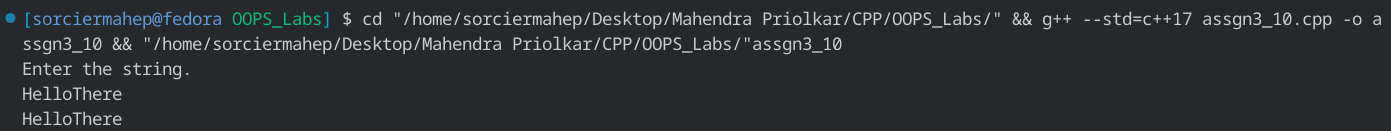
cout << (\*itr);

cout << endl;

return 0;

}

OUTPUT:-



Q6)a)

CODE:-

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v(10, 0);

v[0] = 100;

v[9] = 200;

for (int i = 1; i < 9; i++)

v[i] = 10;

for (int i = 0; i < 10; i++)

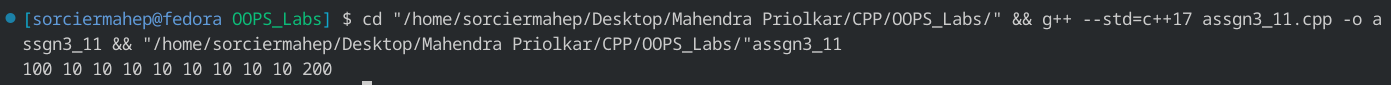
cout << v[i] << " ";

cout << endl;

return 0;

}

OUTPUT:-



Q6)b)

CODE:-

#include <iostream>

#include <list>

using namespace std;

int main()

{

list<int> L1;

L1.push\_front(1); // Put in front

L1.push\_front(2);

L1.push\_back(3);

L1.push\_back(4); // Put in back

cout << "Original list: ";

for (int x : L1)

{

cout << x << " ";

}

cout << endl;

list<int> L2;

list<int>::iterator it;

it = L1.begin();

advance(it, 2);

L2.push\_back(5);

L2.push\_back(6);

L2.push\_back(7);

L1.splice(it, L2); // Add L2 elems to any position of L1 list

cout << "After splice: ";

for (int x : L1)

{

cout << x << " ";

}

cout << endl;

L1.sort(); // Sort L1 list

cout << "After sort: ";

for (int x : L1)

{

cout << x << " ";

}

cout << endl;

list<int> L3;

L3.push\_back(8);

L3.push\_back(9);

L3.push\_back(10);

L1.merge(L3); // Merge sorted lists

cout << "After merge: ";

for (int x : L1)

{

cout << x << " ";

}

cout << endl;

return 0;

}

OUTPUT:-

