**\_\_\_\_\_**

1. Tracing the output.

Assume the address of nValue is equal to 2333

int main(){

double nValue = 7;

double \*pnPtr = &nValue;

cout << pnPtr << endl;

cout << pnPtr+1 << endl;

cout << pnPtr+2 << endl;

cout << pnPtr+3 << endl;

}

1. Find the errors

int nValue = 5;

double dValue = 7.0;

int \*nPtr = &nValue;

double \*dPtr = &dValue;

nPtr = &dValue;

dPtr = &nValue;

1. Trace the following program and write the output:

#include<iostream>

using namespace std;

///////////////////////////////////

class Person {

// Data members of person

public:

Person(int x) { cout << "Person::Person(int ) called" << endl; }

};

class Faculty : public Person {

// data members of Faculty

public:

Faculty(int x):Person(x) {

cout<<"Faculty::Faculty(int ) called"<< endl;

}

};

class Student : public Person {

// data members of Student

public:

Student(int x):Person(x) {

cout<<"Student::Student(int ) called"<< endl;

}

};

class TA : public Faculty, public Student {

public:

TA(int x):Student(x), Faculty(x) {

cout<<"TA::TA(int ) called"<< endl;

}

};

void main() {

TA ta1(30);

}

1. Trace the following program and Find Errors and correct them:

#include <iostream>

using namespace std;

class Base

{ int m;

public:

Base(int mValue){ m = mValue; cout<<"Constructing base"; }

};

class Derived: public Base

{double n;

public:

Derived(double nValue) { n =nValue; cout<<"Constructing derived"; }

};

int main(){

Base b1;

Base b2(3.0);

Derived d1

Derived d2(4.0);

return 0;

}