**Assignment 1 (requires concept of upcasting, dynamic binding, virtual function):**

**===============================================================================**

class A

{

public:

virtual void fn(int i) { cout<<”A::fn(int)”<<endl;}

virtual void gn(double d) { cout<<”A::gn(double)”<<endl;}

int hn(A\* a) { cout<<”A::hn(A\*)”<<endl;}

};

class B : public A

{

public:

void fn(int i) { cout<<”B::fn(int)”<<endl;}

virtual int hn(B\* a) { cout<<”B::hn(B\*)”<<endl;}

};

class C : public B

{

public:

void gn(double d) { cout<<”C::gn(double)”<<endl;}

int hn(B\* a) { cout<<”C::hn(**B**\*)”<<endl;}

};

int main(void)

{ A a; B b; C c;

A \*pA; B \*pB;

}

Read the above code snippet where class B isA type of class A. Fill up the following Table A & B cells (marked ??) that lists different possible combination of accessing the member functions via the base class pointers. Note that if there is an error corresponding to a cell entry mention it as error and specify the reason for that error

**Table A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Initialization** | | |
| **Invocation** |  | pA=&a | pA=&b | pA=&c |
|  |  |  |  |  |
| pA->fn(3) |  | A::fn(int) | B::fn(int) | B::fn(int) |
| pA->gn(3.5) |  | A::gn(double) | A::gn(double) | C::gn(double) |
| pA->hn(&a) |  | A::hn(A\*) | Error2 | Error2 |
| pA->hn(&b) |  | A::hn(A\*) | B::hn(B\*) | B::hn(**B**\*) |

**Table B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Initialization** | | |
| **Invocation** |  | pB=&a | pB=&b | pB=&c |
|  |  |  |  |  |
| pB->fn(3) |  | Error3 | B::fn(int) | B::fn(int) |
| pB->gn(3.5) |  | Error3 | A::gn(double) | C::gn(double) |
| pB->hn(&a) |  | Error3 | Error2 | Error2 |
| pB->hn(&b) |  | Error3 | B::hn(B\*) | B::hn(**B**\*) |

**Error2 :** As we can’t refer A’s object to B pointer, this error prevails.(DownCasting)

**Error3 :** This error is same as the above, we can’t assign A’s object to B pointer.