HINDUSTAN INSTITUTE OF TECHNOLOGY AND SCIENCE

INTERNAL EXAMINATION-2

NAME :N.Appsara

ROLL NO: 19113033

SUBJECT CODE:CSB4118

SUBJECT NAME: OBJECT ORIENTED PROGRAMMING USING

C++

----------------------------EXAM KEY STARTS FROM HERE----------------------------

PART-A

1. (i) The member accessor or dot operator (.)

(ii) The scope resolution operator (::)

(iii) The conditional operator(:?)

(iv) The point to member operator(->)

2) c) Compiler Error in line ” Derived \*dp = new Base;”

3)If virtual keyword is not used the class may get two copies of data members from parent class. This can be resolved by using 'Virtual'. Multipath Inheritance in C++ is derivation of a class from other derived classes, which are derived from the same base class.

This type of inheritance involves other inheritance like multiple, multilevel, hierarchical.

The problem that arises when using multipath inheritance refers to a derived class having more than one parent class that defines property or methods with the same name.

This conflict can be resolved by

i) The order in which the superclasses or parents are given define which property or method will be accessible by the conflict-causing name; the others will remain hidden.

ii) ii) The subclass must resolve the problem by providing a property or method with the name, and by defining how to use the ones from its parents, resolve the ambiguity with the class and a scope resolution operator.

4) class Base //base class

{

public:

virtual void show()=0; //pure virtual function

};

5) exception handler or catch block is the statement used to catch the exceptions thrown by the member function.

Syntax : class()

PART-B

6) #include<iostream>

using namespace std;

class arithmetic\_operators

{

public:

char op;

float a,b;

arithmetic\_operators()

{

this->num2 = 0;

this->num2 = 0;

}

arithmetic\_operators(int a, int b)

{

this->num1 = a;

this->num2 = b;

}

friend arithmetic\_operators operator+(arithmetic\_operators&, arithmetic\_operators&);

};

int main()

{

char op;

float num1, num2;

cout << "Enter operator either + or - or \* or /: ";

cin >> op;

cout << "Enter two operands: ";

cin >> num1 >> num2;

switch(op)

{

case '+':

cout << num1+num2;

break;

case '-':

cout << num1-num2;

break;

default:

cout << "Error! operator is not correct";

break;

}

return 0;

}

7.If derived class defines same function as defined in its base class, it is known as function overriding.It is used to achieve runtime polymorphism. It enables you to provide specific implementation of the function which is already

provided by its base class.

Program :

#include<iostream>

using namespace std;

class Actors

{

public:

void Family()

{

cout << "This is parent class" << endl;

}

};

class Allu : public Actors

{

public:

void Family()

{

cout << "Allu\_Arjun,Allu\_Sirish" << endl;

}

};

class Konidela: public Actors {

public:

void Family()

{

cout << "Ram\_charan,Pawan\_kalyan" << endl;

}

};

class Daggubati : public Actors {

public:

void Family()

{

cout << "Rana\_Daggubati,Venkatesh\_Daggubati" << endl;

}

};

int main()

{

Allu a;

Konidela k;

Daggubati d;

a.Family();

k.Family();

d.Family();

return 0;

}

8) Base class constructors are always called in the derived class constructors. Whenever you create derived class object, first the base class default constructor is executed and then the derived class's constructor finishes execution.

1. Whether derived class's default constructor is called or parameterised is called, base class's default constructor is always

called inside them.

2. To call base class's parameterised constructor inside derived class's parameterised constructor, we must mention it explicitly

while declaring derived class's parameterized constructor.

Program:

class Base

{

int x;

public:

Base(int i)

{

x = i;

cout << "Base Parameterized Constructor\n";

}

};

class Derived : public Base

{

int y;

public:

Derived(int j):Base(j)

{

y = j;

cout << "Derived Parameterized Constructor\n";

}

};

int main()

{

Derived d(10);

}

9) Runtime Polymorphism:

Function overriding is an example of Runtime polymorphism. When child class declares a method, which is already present in the parent class then this is called function overriding, here child class overrides the parent class. This is also known as dynamic or late binding.

Example Program:

#include<iostream>

using namespace std;

class A

{

public:

void disp()

{

cout<<"Super Class Function"<<endl;

}

};

class B:

public A

{

public:

void disp()

{

cout<<"Sub Class Function";

}

};

int main()

{

A obj;

obj.disp();

B obj2;

obj2.disp();

return 0;

}

10) #include<iostream>

using namespace std;

float Divide( float num1, float num2);

{

return (num1/ num2);

}

int main ()

{

float num1 = 123.4;

float num2 = 0;

float result;

result = Divide (num1, num2);

cout << "The quotient of 123.4/0 is"<< result << endl;

}

PART-C

11)

#include<iostream>

using namespace std;

class football

{

char name[30];

float height,weight;

public:

void get()

{

cout<<"Enter name:"<<"\n";

cin>>name;

cout<<"Enter height in metres:"<<"\n";

cin>>height;

cout<<"Enter weight in kilogram:"<<"\n";

cin>>weight;

}

void print()

{

cout<<"Name:"<<name<<"\n";

cout<<"Height:"<<height<<"\n";

cout<<"Weight:"<<weight<<"\n";

}

};

class cricket:public football

{

public:

void get()

{

football::get();

}

void print()

{

football::print();

}

};

class volleyball:public cricket

{

public:

void get()

{

cricket::get();

}

void print()

{

cricket::print();

}

};

int main()

{

int game,n,i;

cout<<"Enter the number if football-1/cricket-2/volleyball-3\n";

cin>>game;

switch(game)

{

case 1:

football f[100];

cout<<"Enter the number of players"<<"\n";

cin>>n;

cout<<"Enter details of "<<n<<" players\n";

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

{

f[i].get();

f[i].print();

}

break;

case 2:

cricket c[100];

cout<<"Enter the number of players"<<"\n";

cin>>n;

cout<<"Enter details of "<<n<<" players\n";

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

{

c[i].get();

c[i].print();

}

break;

case 3:

volleyball v[100];

cout<<"Enter the number of players"<<"\n";

cin>>n;

cout<<"Enter details of "<<n<<" players\n";

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

{

v[i].get();

v[i].print();

}

break;

default: cout<<"Enter correct number\n";

}}