Q1. Create a class FLOAT that contains one float data member. Overload all the four arithmetic operators so that they can operate on the objects of FLOAT.

#include<iostream>

using namespace std;

class FLOAT{

private:

float a;

public:

FLOAT(){

a = 0.0f;

}

void inputData(void);

float display(void);

FLOAT operator+(FLOAT);

FLOAT operator-(FLOAT);

FLOAT operator\*(FLOAT);

FLOAT operator/(FLOAT);

};

int main()

{

FLOAT f1, f2, ans;

cout<<"Addition"<<endl;

f1.inputData();

f2.inputData();

ans = f1 + f2;

cout<<"f1 + f2 = "<<ans.display()<<endl;

cout<<"-------------------------------"<<endl;

cout<<"Subtraction"<<endl;

f1.inputData();

f2.inputData();

ans = f1 - f2;

cout<<"f1 - f2 = "<<ans.display()<<endl;

cout<<"-------------------------------"<<endl;

cout<<"Multiplication"<<endl;

f1.inputData();

f2.inputData();

ans = f1 \* f2;

cout<<"f1 \* f2 = "<<ans.display()<<endl;

cout<<"-------------------------------"<<endl;

cout<<"Division"<<endl;

f1.inputData();

f2.inputData();

ans = f1 / f2;

cout<<"f1 / f2 = "<<ans.display()<<endl;

cout<<"-------------------------------"<<endl; }

void FLOAT::inputData(void){

cout<<"Enter a : ";

cin>>a;

}

float FLOAT::display(void){

return a;

}

FLOAT FLOAT::operator+(FLOAT tmp){

tmp.a = a + tmp.a;

return tmp;

}

FLOAT FLOAT::operator-(FLOAT tmp){

tmp.a = a - tmp.a;

return tmp;

}

FLOAT FLOAT::operator\*(FLOAT tmp){

tmp.a = a \* tmp.a;

return tmp;

}

FLOAT FLOAT::operator/(FLOAT tmp){

tmp.a = a / tmp.a;

return tmp;

}

Q2. Define a class Rectangle and overload area function for different types of data type.

#include <iostream>

using namespace std;

class Rectangle

{

public:

int area(int, int);

float area(float, float);

float area(int, float);

};

int main()

{

Rectangle r;

cout<<"area (int, int) = "<<r.area(5, 9)<<endl;

cout<<"area (float, float) = "<<r.area(6.78f, 12.67f)<<endl;

cout<<"area (int, float) = "<<r.area(6, 9.45f)<<endl;

}

int Rectangle::area(int l, int w){ return l \* w; }

float Rectangle::area(float l, float w){ return l \* w; }

float Rectangle::area(int l, float w){ return l \* w; }

Q3. Define a base class Animals having member function sound() . Define another derived class from Animals class named Dogs. You need to override the sound function of the base class in the derived class.

#include <iostream>

using namespace std;

class Animals{

public:

void sound(void);

};

class Dogs:public Animals{

public:

void sound(void);

};

int main(){

Dogs d;

d.sound();

}

void Animals::sound(void){ cout<<"Animal class"<<endl; }

void Dogs::sound(void){ cout<<"Dogs class"<<endl; }

Q4. Define a class Addition that can add 2 or 3 numbers of different data types using function overloading.

#include <iostream>

using namespace std;

class Addition{

public:

void add(int, int);

void add (int, float, float);

};

int main(){

Addition obj1, obj2, obj3;

obj1.add(5, 5);

obj2.add(6, 4.5f, 9);

obj3.add(5, 4);

}

void Addition::add(int a, int b){

cout<<"addition is >>> "<<a + b<<endl;

}

void Addition::add(int a, float b, float c=0){

cout<<"addition is >>> "<<a + b + c<<endl;

}

Q5. Define a class A having multiple constructors. Define another class B derived from class A. Create derived class constructors and show use of constructor in this single inheritance.

#include <iostream>

using namespace std;

class A{

private:

int a;

public:

A(){ a = 10; cout<<"Class a default constructor called"<<endl; }

};

class B:public A{

private:

int b;

public:

B(){

b = 20;

cout<<"class b default constructor called"<<endl;

}

B(int tmp){

b = tmp;

cout<<"parameterized constructor called"<<endl;

}

B(B &tmp){

b = tmp.b;

cout<<"copy constructor called"<<endl;

}

void disp(void){

cout<<b<<endl;

}

};

int main(){

B b1;

b1.disp();

cout<<"-----------------"<<endl;

B b2(50);

b2.disp();

cout<<"-----------------"<<endl;

B b3(b2);

b3.disp();

}

Q6. C++ Program to illustrate the use of Constructors in multilevel inheritance of your choice.

#include <iostream>

using namespace std;

class A{

private:

int a;

public:

A(){a=10;}

};

class B:public A{

private:

int b;

public:

B(){b=20;}

int getB(void)

{

return b;

}

};

class C: public B{

private:

int c;

public:

C()

{

c=30;

cout<<"default constructor called"<<endl;

cout<<c<<endl<<"--------------------------------"<<endl;

}

C(int tmp)

{

c = tmp;

cout<<"parameterized constructor called"<<endl;

cout<<c<<endl<<"--------------------------------"<<endl;

}

C(B &tmp)

{

c = tmp.getB();

cout<<"copy constructor called"<<endl;

cout<<c<<endl<<"--------------------------------"<<endl;

}

};

int main()

{

B b1;

C c1, c2(100), c3(b1);

}

Q7. C++ Program to illustrate the use of Constructors in single inheritance of your choice.

#include <iostream>

using namespace std;

class Base{

private:

int \*a;

public:

Base()

{

a = (int \*)malloc(sizeof(int));

\*a = 10;

}

int getData(void)

{

return \*a;

}

int disp(void)

{

cout<<"Address of a: "<<a<<" and data of a: ";

return \*a;

}

};

class Derived:public Base{

private:

int \*b;

public:

Derived()

{

b = (int \*)malloc(sizeof(int));

\*b = 20;

}

Derived(Base &tmp)

{

b = (int \*)malloc(sizeof(int));

\*b= tmp.getData();

}

int disp(void)

{

cout<<"Address of b: "<<b<<" and data of b: ";

return \*b;

}

};

int main()

{

Base b1;

cout<<b1.disp()<<endl;

Derived d1(b1);

cout<<d1.disp()<<endl;

}

Q8. Write a C++ program to find the factorial of a number using copy constructor

#include <iostream>

using namespace std;

class Factorial

{

private:

int num;

public:

Factorial()

{

cout<<"Enter number : ";

cin>>num;

}

Factorial(Factorial &tmp)

{

cout<<"copy constructor called"<<endl;

cout<<"factorial of "<<tmp.num;

num = tmp.fact(tmp.num, tmp.num-1);

cout<<" is "<<num;

}

int fact(int first, int second)

{

if(second > 1)

{

first = first \* second;

first = fact(first, second - 1);

return first;

}

else

{

return first;

}

}

};

int main()

{

Factorial f1, f2(f1);

}

Q9.Write a C++ program to calculate the area of triangle, rectangle and circle using

constructor overloading. The program should be menu driven.

#include <iostream>

#include <cstdlib>

using namespace std;

class Shapes

{

public:

Shapes(){}

Shapes(float b, float h)

{

cout<<"Area of triangle : "<<b \* h<<endl;

}

Shapes(float w, float l, int dummy)

{

cout<<"Area of rectangle : "<<w \* l<<endl;

}

Shapes(float r)

{

cout<<"Area of Circle : "<<3.14 \* r \*r<<endl;

}

};

int main()

{

int choice = 0, reset =0;

do{

cout<<"1. Area of Triangle"<<endl;

cout<<"2. Area of Rectangle"<<endl;

cout<<"3. Area of Circle"<<endl<<endl;

cout<<"Enter choice : ";

cin>>choice;

switch(choice)

{

case 1:

{

Shapes Aot(5.0f, 9.6f);

break;

}

case 2:

{

Shapes AoR(10.5f, 15.0f, 0);

break;

}

case 3:

{

Shapes AoC(7.2f);

break;

}

default:

{

cout<<"Wrong choice"<<endl;

break;

}

}

cout<<"Do you want to retry ? 1/0 : ";

cin>>reset;

system("cls");

}while(reset == 1);

}

Q10. Create a C++ class for player objects with the following attributes: player no., name,

number of matches and number of goals done in each match. The number of matches varies for each player. Write a parameterized constructor which initializes player no., name, creates an array for number of goals and number of matches dynamically.

#include <iostream>

#include <string>

using namespace std;

class Player

{

private:

int player\_no;

string name;

int \*matches;

int \*goals;

public:

Player(){}

Player(int player\_no, string name, int total\_match)

{

this->player\_no = player\_no;

this->name = name;

this->matches = (int \*)malloc(sizeof(int) \* total\_match);

this->goals = (int \*)malloc(sizeof(int) \* total\_match);

for(int i=0; i<total\_match; i++)

{

cout<<"Enter match number : ";

cin>>matches[i];

cout<<"Enter goal associate with match : ";

cin>>goals[i];

}

}

};

int main()

{

Player p1(101, "yash", 3);

}