/\*1. Write a C++ program to read the data of N employees and compute the Net salary of each employee (DA=52% of Basic and Income Tax (IT) = 30% of the gross salary). For that, create an Employee class with Employee number, Employee name, Basic, DA, IT, Net Salary.(Concept: Array of Objects)\*/

#include <iostream>

using namespace std;

class employee{

private:

char ename[20];

int enume,basic;

float da,it,net;

public:

void read(){

cout<<"enter your name:"<<endl;

cin>>ename;

cout<<"enter your id:"<<endl;

cin>>enume;

cout<<"enter basic salary:"<<endl;

cin>>basic;

}

void netsalary(){

da=0.52\*basic;

it=(basic+da)\*0.3;

net=(da+basic)-it;

cout<<"net salary:"<<net<<endl;}};

int main()

{employee e[10];

int n;

cout<<"enter value of n:"<<endl;

cin>>n;

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

cout<<"enter details of employee"<<i+1<<endl;

e[i].read();e[i].netsalary();}return 0;}

/\*2. Write a C++ program with two classes ABC and XYZ with one integer data member in each class. Write member functions to read and display, place a friend function called max() in these classes which takes the data members of these classes and computes a maximum of two data members.

(Concept: Friend function and Reference variable.)

\*/

#include <iostream>

using namespace std;

class XYZ;

class ABC{

private: int a;

public:

void read(){

cout<<"enter value of number 1:"<<endl;

cin>>a;

}

void display(){

cout<<"value of a:"<<a<<endl;

}

friend void maxs(ABC,XYZ);};

class XYZ{

private: int b;

public:

void read(){

cout<<"enter value of number 2:"<<endl;

cin>>b;

}

void display(){

cout<<"value of b:"<<b<<endl;

}

friend void maxs(ABC,XYZ);};

void maxs(ABC m,XYZ n){

int l;

l=max(m.a,n.b);

cout<<"max is:"<<l<<endl;

}

int main()

{

ABC abc;

XYZ xyz;

abc.read();

xyz.read();

maxs(abc,xyz);

return 0;

}

/\*3. Write a program to find the largest, smallest & second largest of three numbers.

(use inline function MAX and MIN to find largest & smallest of 2 numbers)

(Concept: Inline function)\*/

#include <iostream>

using namespace std;

inline int MAX(int a,int b){

int l=max(a,b);

return l;

}

inline int MIN(int a,int b){

int l=min(a,b);

return l;

}

int main()

{int a,b,c;

cout<<"enter a and b and c:"<<endl;

cin>>a>>b>>c;

cout<<"Maximum of 3 elements is:"<<MAX(MAX(a,b),c)<<endl;

cout<<"Minimum of 3 elements is:"<<MIN(MIN(a,b),c)<<endl;

int sl,l,s;

l=MAX(MAX(a,b),c);

s=MIN(MIN(a,b),c);

if(l==a && s==b){

cout<<"sl is:"<<c<<endl;

}

else if(l==b && s==c){

cout<<"sl is:"<<a<<endl;

}

else{

cout<<"sl is:"<<b<<endl;

}

return 0;

}

4 /\*4. Write a Program to design a class having a static member function named ShowCount()

which has the property of displaying the number of objects created of the class.

(Concept: Applications of the Static keyword)

\*/

#include <iostream>

using namespace std;

class myclass{

private:

static int counts;

public:

myclass(){

counts++;}

static void showcount(){

cout<<"the value of count after creating obj is:"<<counts<<endl;

}};

int myclass::counts=0;//imp

int main()

{myclass c1;

myclass c2;

myclass c3;

myclass::showcount();

return 0;

}

/\*5. Write a C++ program to find the volume of a cylinder, sphere and cube.

(Concept: Function overloading)

\*/

#include <iostream>

using namespace std;

class sol{

public:

float volume(int r,int h){

return (3.14\*r\*r\*h);}

float volume(int a){

return (a\*a\*a);

}

float volume(float r){

return (3.14\*r\*r\*r\*4)/3;}};

int main()

{sol s;

int a,r,h;

float R,v1,v2,v3;

cout << "enter the dimensions of cube,cylinder and sphere:" << endl;

cout<<"enter the side of a cube:"<<endl;

cin>>a;

cout<<"enter the radius and height of cylinder:"<<endl;

cin>>r>>h;

cout<<"enter the radius of sphere:"<<endl;

cin>>R;

v1=s.volume(r,h);

v2=s.volume(R);

v3=s.volume(a);

cout<<"volume are:"<<v3<<" "<<v2<<" "<<v1<<endl;

return 0;

}

/\*Write a C++ program to demonstrate the working of a copy constructor. Implement a class called Point

with private data members X and Y as the points and getX() and getY() are the getter functions to get the values and

print the same using the main() function.

(Concept: Copy constructor)\*/

#include <iostream>

using namespace std;

class point{

private:

int x,y;

public:

point(int a,int b){

x=a;

y=b;

}

point(point& p1){

x=p1.x;

y=p1.y;}

int getx(){

return x;}

int gety(){

return y;}

};

int main()

{point p1(5,6),p2=p1;

cout << "x and y are:"<<p1.getx()<<p1.gety()<<p2.getx()<<p2.gety()<< endl;

return 0;

}

/\*7. Write a C++ program to create a class Data with integer, character and float data members.

Demonstrate Constructor Overloading on this class with all types of constructors including default argument constructor.

(Concept: Constructor Overloading)

\*/

#include <iostream>

using namespace std;

class data{

public:

int a;

char b;

float c;

data(){

a=1;

b='a';

c=3.44;}

data(int x,char y,float z){

a=x;

b=y;

c=z;

}

data(int x,float z,char y='g'){

a=x;

c=z;

b=y;}

data(data& d){

a=d.a;

b=d.b;

c=d.c;}};

int main()

{data d1;

int u;

char f;

float o;

cout << "write int ,char float:" << endl;

cin>>u>>f>>o;

data d2(u,f,o);

data d3(u,o);

data d4=d2;

cout<<" int char float values are:"<<d1.a<<" "<<d1.b<<" "<<d1.c<<" "<<d2.a<<" "<<d2.b<<" "<<d2.c<<" "<<d3.a<<" "<<d3.b<<" "<<d3.c<<" "<<d4.a<<" "<<d4.b<<" "<<d4.c<<endl;

return 0;

}

/\*8. Write a C++ program to apply bubble sort on an array of integers and float using the concept of function template.

(Concept: Class Template)\*/

#include <iostream>

using namespace std;

template <typename T>void sorting(T a[],int n){

for(int i=0;i<n-1;i++){

for(int j=0;j<n-i-1;j++){

if(a[j]>a[j+1]){

swap(a[j],a[j+1]);}}}

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

cout<<a[i]<<" "<<endl;;

}}

int main(){int n;

cout<<"enter n:"<<endl;

cin>>n;

int a[n];

float b[n];

cout << "enter the elements of int array:" << endl;

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

cin>>a[i];

}

sorting <int> (a,n);

cout << "enter the elements of array:" << endl;

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

cin>>b[i];

}

sorting <float> (b,n);

return 0;

}

/\*9. Write a C++ program to read and print employee information using multiple inheritance. Create 2 base classes namely BasicInfo and DeptInfo which contains getBasicInfo() and getDeptInfo() respectively to read the information.

(Concept: multiple inheritance.)

\*/

#include <iostream>

using namespace std;

class basicinfo{

protected:

char name[30];

int id;

char gender;

public:

void getbasicinfo(){

cout<<"enter your basic info,name,id,gender"<<endl;

cin.getline(name,30);

cin>>id;

cin>>gender;

cin.ignore();

}};

class deptinfo{

protected:

char deptname[10];

int deptid;

public:

void getdeptinfo(){

cout<<"enter your dept info:"<<endl;

cin.getline(deptname,10);

cin>>deptid;

}};

class employee : public basicinfo,public deptinfo{

public:

void det(){

cout<<"your name:"<<name<<"\nyour id"<<id<<"\nyour gender :"<<gender<<"\nyour dept name:"<<deptname<<"\nyour deptid:"<<deptid<<endl;

}};

int main()

{employee e;

e.getbasicinfo();

e.getdeptinfo();

e.det();

return 0;

}

/\*10. Write a C++ program to demonstrate the uses of constructors in derived class concepts. The three classes that can be created are Alpha, Beta and Gamma in this order having an “is-a” relationship. Create at least one data member and one member function in each class. That is n1 and putAlpha() in Alpha class, n2 and putBeta() in Beta class, n3 and putGamma() in Gamma class.

(Concept: Use of Constructors in Derived Classes.)

\*/

#include <iostream>

using namespace std;

class alpha{

protected:

int n1;

public:

alpha(int x){

n1=x;

}

void putalpha(){

cout<<"alpha is contructed"<<endl;

cout<<"value of x:"<<n1<<endl;}

};

class beta{

protected:

int n2;

public:

beta(int y){

n2=y;

}

void putbeta(){

cout<<"beta is contructed"<<endl;

cout<<"value of y:"<<n2<<endl;}

};

class gamma : public alpha,public beta{

protected:

int n3;

public:

gamma(int x,int y,int z):alpha(x),beta(y)//imp{

{n3=z;

}

void putgamma(){

cout<<"gamma is contructed"<<endl;

cout<<"value of z:"<<n3<<endl;}};

int main()

{

gamma g(2,3,4);

g.putalpha();

g.putbeta();

g.putgamma();

return 0;}