**Lab – 7**

Operator overloading problems

1. **Create a class Time with three private variables int h,m,s; Create a function to overload ‘+’ operator to add two time variables.**

**int main(){**

**Time t1(5,15,34),t2(9,53,58),t3;**

**t3 = t1 + t2; t3.show();**

}

#include<iostream>

using namespace std;

class time{

int h,m,s;

public:

time(int a, int b, int c){

h=a;

m=b;

s=c;

}

time(){}

time operator+(time obj){

time temp;

temp.h=this->h+obj.h;

temp.m=this->m+obj.m;

temp.s=this->s+obj.s;

if(temp.s){

temp.m+=temp.s/60;

temp.s=temp.s%60;

}

if(temp.m>60){

temp.h+=temp.m/60;

temp.m=temp.m%60;

}

return temp;

}

void show(){

cout<<"hours: "<<h<<"minutes: "<<m<<"seconds: "<<s<<endl;

}

};

int main(){

time t1(5,15,34),t2(9,53,58),t3;

t3=t1+t2;

t3.show();

return 0;

}

1. **Define a class STRING and overload = = to compare two strings and + operator for concatenation of two strings.**

#include<iostream>

#include<string.h>

using namespace std;

class STRING{

char str1[20];

public:

STRING(char str\_1[20]){

strcpy(str1,str\_1);

}

STRING operator==(STRING obj){

int result;

result=strcmp(this->str1, obj.str1);

cout<<"Result: "<<result<<endl;

}

STRING operator+(STRING obj){

strcat(this->str1,obj.str1);

cout<<"Concatinate d str is: "<<this->str1;

}

};

int main(){

STRING obj1("hello"),obj2("world");

obj1==obj2;

obj1=obj1+obj2;

return 0;

}

1. **Write a program in C++ to create a class matrix and overload \* operator using friend function to multiply two matrices.**

#include<iostream>

#include<math.h>

using namespace std;

class test{

int a[2][2];

public:

void getval(){

int i,j;

for(i=0;i<2;i++){

for(j=0;j<2;j++){

cin>>a[i][j];

}

}

}

void display(){

int i,j;

for(i=0;i<2;i++){

for(j=0;j<2;j++){

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

}

cout<<endl;

}

}

friend test operator\*(test &m, test &n);

};

test operator\*(test &m, test &n){

int i,j,k;

test o;

for(i=0;i<2;i++){

for(j=0;j<2;j++){

o.a[i][j]=0;

for(k=0;k<2;k++){

o.a[i][j]=o.a[i][j]+m.a[i][k]\*n.a[k][j];

}

}

}

return o;

}

int main(){

test t1,t2,t3;

t1.getval();

t2.getval();

t3=t1\*t2;

t3.display();

return 0;

}

1. **Overload ‘[]’ to check array index out of bounds problem at run time.**

#include<iostream>

using namespace std;

//const int max=10;

class array{

int arr[10];

public:

array(){

int i;

for(i=0;i<10;i++){

arr[i]=i+1;

}

}

int &operator[](int index);

};

int& array::operator[](int index){

if(index>10)

{

cout<<"arr out of bound"<<endl;

exit(0);

}

return arr[index];

}

int main(){

array a;

cout<<a[2]<<endl;

cout<<a[12]<<endl;

return 0;

}

1. **Overload ‘()’ to input arbitrary number of input arguments for an object.**

#include<iostream>

using namespace std;

class test{

int x;

int y;

public:

test(){

x=2;

y=3;

}

test operator()(int a, int b, int c){

test m;

m.x=a+b;

m.y=a+c;

return m;

}

void display(){

cout<<"x: "<<x<<" "<<"y: "<<y<<endl;

}

};

int main(){

test obj1, obj2;

obj1.display();

obj2=obj1(1,1,1);

obj2.display();

return 0;

}

1. **Write a program in C++ to overload input operator (>>) and output operator (<<).**

#include<iostream>

using namespace std;

class frac{

int n;

int d;

public:

frac(){

n=0;

d=1;

}

friend istream& operator>>(istream& cin, frac& x){

cin>>x.n>>x.d;

return cin;

}

friend ostream& operator<<(ostream& cout, frac& x){

cout<<x.n<<" / "<<x.d<<endl;

return cout;

}

};

int main(){

frac f;

cout<<"Enter frac: "<<endl;

cin>>f;

cout<<f;

return 0;

}

1. **Write a program to convert basic data type (float) to user defined data type (object).**

**class Test {**

**private: //….**

**public:**

**Test ( data\_type) { // conversion code }**

**};**

#include<iostream>

using namespace std;

class test{

float value;

public:

test(float x){

value=x;

}

void display(){

cout<<"val in obj: "<<value;

}

};

int main(){

float float\_val=12.34;

test obj1(float\_val);

obj1.display();

return 0;

}

1. **Write a program to convert UDT to basic data type (float)**

**class Test{**

**public:**

**operator data\_type() { //Conversion code }**

**};**

#include<iostream>

using namespace std;

class test{

float value;

public:

test(float x){

value=x;

}

float get\_val(){

return value;

}

};

int main(){

test obj1(12.34);

float x;

x=obj1.get\_val();

cout<<"value of basic float: "<<x<<endl;

return 0;

}

1. **How will you convert one UDT to another UDT. For example conversion of polar to cartesian system.**

**Polar p(10,5);**

**Cartesian c = p;**

**c.show();**

#include<iostream>

#include<cmath>

using namespace std;

class polar{

float r;

float theta;

public:

polar(float a, float b){

r=a;

theta=b;

}

float get\_r(){

return r;

}

float get\_theta(){

return theta;

}

};

class cart{

float x;

float y;

public:

cart(float a, float b){

x=a;

y=b;

}

float get\_x(){

return x;

}

float get\_y(){

return y;

}

cart operator=(polar p){

x=p.get\_r()\*cos(p.get\_theta());

y=p.get\_r()\*sin(p.get\_theta());

return cart(x,y);

}

};

int main(){

polar p1(1,M\_PI\_4);

cart c1(0,0);

c1=p1;

cout<<"x: "<<c1.get\_x()<<" "<<"y: "<<c1.get\_y();

return 0;

}