1. What is the difference between a (binary) operator and a function?

A binary operator is an operator that operates on two operands and manipulates them to return a result. Binary operators are presented in the form: Operand1 Operator Operand2. A function is statement that performs a task, every c++ program has at least one function which is main(). The difference is operators are used to produce a result in a selection statement of true or false and are placed inside a function, the function performs the whole task.

1. Is it possible using operator overloading to change the effect of + on integers? Why or why not?

No it is not possible to overload the plus operator for ints, when you overload the operator+ one of the two parameters has to be a user defined class/struct. Otherwise the program won’t compile. You will get the error ”int operator+(int, int) must have an argument of class or enumerated type”

1. Why can’t we overload << or >> as member operators?

Because you’re not allowed to modify ostream, if you overload as a member function a << b is interpreted as a.operator<<(b) so it only takes one explicit parameter. This requires that the overload be part of of the class used as the left hand operand, it’s not useful with normal ostreams and such, It would require that your overload be part of the ostream class, not part of your class. Since you’re not allowed to modify ostream you can’t perform this.

4. Below is the definition for a class called Percent. Objects of type Percent represent percentages such as 10% or 99%. Give the definitions of the overloaded operators >> and << so that they can be used for input and output with objects of the class Percent. Assume that input always consists of an integer followed by the character ’%’, such as 25%. All percentages are whole numbers and are stored in an int member variable named value. You do not yet need to define the other overloaded operators and do not yet need to define the constructor. Firstly, you only have to define the overloaded operators >> and <<.

#include <iostream>

using namespace std;

class Percent

{

public:

friend bool operator ==(const Percent& first,

const Percent& second);

friend bool operator <(const Percent& first,

const Percent& second);

Percent( );

friend istream& operator >>(istream& inputStream,

Percent& aPercent);

friend ostream& operator <<(ostream& outputStream,

const Percent& aPercent);

//There will be other members and friends.

private:

int value;

};

1. Add to the class overloaded operators to add, subtract and multiply 2 percentages. If you have to re-write the code so that wherever possible the overloaded operators are made member functions of the class.

friend const Percent operator+(const Percent& first, const Percent& second);

friend const Percent operator\*(const Percent& first, const Percent& second);

friend const Percent operator-(const Percent& first, const Percent& second);

1. Implement all of the overloaded operators in the class (and any stand-alone ones still declared as friend functions of the class)

#include "Percent.h"

#include <iostream>

using namespace std;

Percent::Percent(int i) {

value = i;

}

const Percent operator+(const Percent& first, const Percent& second) {

Percent temp(first.value + second.value);

return temp;

}

const Percent operator\*(const Percent& first, const Percent& second) {

Percent temp(first.value \* second.value);

temp = temp.value / 100;

return temp;

}

const Percent operator-(const Percent& first, const Percent& second) {

Percent temp(first.value - second.value);

return temp;

}

bool operator==(const Percent& first, const Percent& second) {

return first.value == second.value;

}

bool operator<(const Percent& first, const Percent& second) {

if (first.value < second.value) {

return true;

}

else {

return false;

}

}

istream& operator >> (istream& inputStream, Percent& aPercent){

inputStream >> aPercent.value;

return inputStream;

}

ostream& operator << (ostream& outputStream, const Percent& aPercent) {

outputStream << aPercent.value;

return outputStream;

}

**SEE ATTACHED FOLDER LAB7PARTA FOR ANSWER!**

*N.B. Think carefully about the multiplication: 50% \* 50% should mean 50% OF 50%, and therefore provide the answer 25% not 2500%*

1. Write a program which fully tests your class.

#include <iostream>

#include "Percent.h"

using namespace std;

int main() {

Percent p1(40);

Percent p2(20);

Percent p3(0);

cout << "Adding.." << endl;

p3 = p1 + p2;

cout << "Answer = " << p3 << endl;

cout << "Subtracting.." << endl;

p3 = p1 - p2;

cout << "Answer = " << p3 << endl;

cout << "Multiplying.." << endl;

p3 = p1 \* p2;

cout << "Answer = " << p3 << "%" << endl;

p2 = p2 - 20;

cout << "Checking less than.." << endl;

if (p3 < p2) {

cout << "Yes " << p3 << " is less than " << p2 << endl;

}

else {

cout << "No " << p3 << " is not less than " << p2 << endl;

}

system("pause");

}

5. Write a Money class, with data members euro and cent. Add overloaded operators (as member functions where this is possible) to do the following:

* Subtract 2 Money amounts.
* Multiply a Money amount by an integer to facilitate code like this:

myMoney = yourMoney \* 2;

* Divide a Money amount by a decimal
* Output a Money amount using the insertion operator
* Compare 2 Money amounts (with <, > and ==)

**SEE ATTACHED FOLDER LAB7PARTAMONEY FOR ANSWER!**