**HW3 ANSWERS**

1. Define a class named PrimeNumber that stores a prime number. The default constructor should set the prime number to 1. Add another constructor that allows the caller to set the prime number (make sure that the new number is a prime number). Also add a function to get the prime number. Finally, overload the prefix and postfix ++ and -- operators so they return a PrimeNumber object that is the next largest prime number (for ++) and the next smallest prime number (for --). For example, if the object’s prime number is set to 13, then invoking ++ should return a PrimeNumber object whose prime number is set to 17. Create an appropriate test program for the class.

#include <iostream>

using namespace std;

class PrimeNumber{

int prime;

//isPrime is private because I inted

//to use it only in the class

bool isPrime(int);

int nextPrime();

int prevPrime();

public:

PrimeNumber();

PrimeNumber(int);

int getPrime();

PrimeNumber operator++();

PrimeNumber operator++(int);

PrimeNumber operator--();

PrimeNumber operator--(int);

};

PrimeNumber::PrimeNumber():prime(1){}

PrimeNumber::PrimeNumber(int newPrime){

if(isPrime(newPrime))

prime = newPrime;

else

{

cout<<"error"<<endl;

system("pause");

exit(1);

}

}

int PrimeNumber::getPrime(){return prime;}

PrimeNumber PrimeNumber::operator++(){

prime = nextPrime();

return PrimeNumber(prime);

}

PrimeNumber PrimeNumber::operator++(int){

PrimeNumber temp;

temp.prime=prime;

prime=nextPrime();

return temp;

}

PrimeNumber PrimeNumber::operator--(){

prime = prevPrime();

return PrimeNumber(prime);

}

PrimeNumber PrimeNumber::operator--(int){

PrimeNumber temp;

temp.prime=prime;

prime=prevPrime();

return temp;

}

bool PrimeNumber::isPrime(int newPrime){

bool isprime=true;

for(int i=2;i<newPrime&&isprime;i++)

{

if(newPrime%i==0)

isprime=false;

}

return isprime;

}

int PrimeNumber::prevPrime(){

bool stop=false;

int newPrime=prime;

while(!stop && newPrime!=1){

newPrime--;

if(isPrime(newPrime))

stop=true;

}

return newPrime;

}

int PrimeNumber::nextPrime(){

bool stop=false;

int newPrime=prime;

do{

newPrime++;

if(isPrime(newPrime))

stop=true;

}while(!stop);

return newPrime;

}

int main(){

PrimeNumber p1(17),p2,p3;

p2=--p1;

cout<<"p2: "<<p2.getPrime()<<endl;

cout<<"p1: "<<p1.getPrime()<<endl;

p3=p1++;

cout<<"p3: "<<p3.getPrime()<<endl;

cout<<"p1: "<<p1.getPrime()<<endl;

return 0;

}

1. Do Programming Project 6.10, the definition of a Temperature class, except overload ==, << and >>. The == operator should return true if the two temperature values are identical while << should output the temperature and >> should input the temperature. Create appropriate tests for the overloaded operators.

**Note:** The first printing of this question erroneously stated to overload ==, ≪, and ≫ as member operators. As described in the book, this is not really possible to get the desired effect.

Do Programming Project 6.10, the definition of a Temperature class, except overload ==, << and >>. The == operator should return true if the two temperature values are identical while << should output the temperature and >> should input the temperature. Create appropriate tests for the overloaded operators.

**Note:** The first printing of this question erroneously stated to overload ==, ≪, and ≫ as member operators. As described in the book, this is not really possible to get the desired effect.

#include <iostream>

#include <cmath>

using namespace std;

class Temperature

{

public:

Temperature();

Temperature(double initialKelvin);

friend bool operator ==(const Temperature& temp1, const Temperature& temp2);

friend ostream& operator <<(ostream& outputStream, const Temperature& temp);

friend istream& operator >>(istream& inputStream, Temperature& temp);

void setTempKelvin(double temp);

void setTempFahrenheit(double temp);

void setTempCelsius(double temp);

double getTempKelvin();

double getTempFahrenheit();

double getTempCelsius();

private:

double kelvin; // Internally store temp in kelvin

};

Temperature::Temperature() : kelvin(0){}

Temperature::Temperature(double initialKelvin): kelvin(initialKelvin){}

bool operator ==(const Temperature& temp1, const Temperature& temp2)

{

// Shouldn't compare doubles for equality

double diff = abs(temp1.kelvin - temp2.kelvin);

if (diff < 0.001)

return true;

return false;

}

ostream& operator <<(ostream& outputStream, const Temperature& temp)

{

outputStream << temp.kelvin << " degrees Kelvin" << endl;

return outputStream;

}

istream& operator >>(istream& inputStream, Temperature& temp)

{

inputStream >> temp.kelvin;

return inputStream;

}

void Temperature::setTempKelvin(double temp)

{

kelvin = temp;

}

void Temperature::setTempFahrenheit(double temp)

{

kelvin = (5.0 \* (temp - 32) / 9) + 273.15;

}

void Temperature::setTempCelsius(double temp)

{

kelvin = temp + 273.15;

}

double Temperature::getTempKelvin()

{

return kelvin;

}

double Temperature::getTempCelsius()

{

return kelvin - 273.15;

}

double Temperature::getTempFahrenheit()

{

return (getTempCelsius() \* 9.0 / 5) + 32;

}

int main( )

{

Temperature temp;

cout << "Input a temperature in kelvin." << endl;

cin >> temp;

cout << "In Celsius: " << temp.getTempCelsius() << endl;

cout << "In Fahrenheit: " << temp.getTempFahrenheit() << endl;

cout << temp << endl;

cout << endl;

Temperature temp2(100);

Temperature temp3(100);

Temperature temp4(50);

cout << "temp2 == temp3 : " <<

(temp2 == temp3) << endl;

cout << "temp2 == temp4 : " <<

(temp2 == temp4) << endl;

cout << "Enter a character to exit." << endl;

char wait;

cin >> wait;

return 0;

}

1. Write a program that outputs a histogram of grades for an assignment given to a class of students. The program should input each student’s grade as an integer and store the grade in a vector. Grades should be entered until the user enters -1 for a grade. The program should then scan through the vector and compute the histogram. In computing the histogram the minimum value of a grade is 0 but your program should determine the maximum value entered by the user. Use a dynamic array to store the histogram. Output the histogram to the console.

Output the histogram count at the end. For example, if the input grades are 3, 0, 1, 3, 3, 5, 5, 4, 5, 4, 6, -1 then the program should output

1 grade(s) of 0

1 grade(s) of 1

0 grade(s) of 2

3 grade(s) of 3

2 grade(s) of 4

3 grade(s) of 5

0 grade(s) of 6

Output the histogram count at the end. For example, if the input grades are 3, 0, 1, 4, 3, 4, 3, 3, 0, -1 then the program should output

2 grade(s) of 0

1 grade(s) of 1

0 grade(s) of 2

4 grade(s) of 3

2 grade(s) of 4

#include <iostream>

#include <vector>

using namespace std;

int main(){

vector<int> v;

int number,max=0;

do{

cout<<"enter a number -1 to exit: ";

cin>>number;

//error message will be displayed only if is a negative #

//and the number is not -1

if(number < 0 && number != -1)

cout<<"the number can't be negative"<<endl;

else if(number!=-1){

//places number in vector

v.push\_back(number);

//finds the max number of input

//because it will be the size of dynamic array

if(number>max)

max=number;

}

}while(number!=-1);

//if no values entered in vector display error

if(v.size()==0)

{

cout<<"there were no numbers entered"<<endl;

exit(1);

}

//creation of dynamic array

int \*histogram = new int[++max];

//initializes the histogram to 0

for(int i=0;i<max;i++)

histogram[i]=0;

//increments the value of histogram

//using as index the value entered in vector

for(int i=0;i<v.size();i++)

histogram[v[i]]++;

//display of histogram

for(int i=0;i<max;i++)

cout<<histogram[i]<<" grade's of "<<i<<endl;

//delete histogram from heap

delete [] histogram;

histogram=NULL;

return 0;

}

1. Create a class named Student that has three member variables:

name – A string that stores the name of the student

numClasses – An integer that tracks how many courses the student is currently enrolled in

classList – A dynamic array of strings used to store the names of the classes that the student is enrolled in

Write appropriate constructor(s), mutator, and accessor methods for the class along with the following:

 A function that inputs all values from the user, including the list of class names. This method will have to support input for an arbitrary number of classes.

 A function that outputs the name and list of all courses.

 A function that resets the number of classes to 0 and the classList to an empty list.

Write a main function that tests all of your functions.

CodeMate hint: Recall that cin >> variable leaves a newline in the buffer. This can be a problem if you are mixing cin >> variable and getline. Use cin.ignore to discard the newline.

#include <iostream>

#include <string>

using namespace std;

class student{

string name;

int nClasses;

string \*classList;

public:

student();

student(string);

void setName(string);

int getNClasses() const;

string getName() const;

void input();

void ouput();

void reset();

};

student::student(){

setName("not defined");

nClasses=0;

classList = NULL;

}

student::student(string newName){

setName(newName);

nClasses=0;

classList = NULL;

}

void student::setName(string newName){

name = newName;

}

int student::getNClasses() const{return nClasses;}

string student::getName() const{return name;}

void student::input(){

//reset in the case the function is

//called more than one time

reset();

cout<<"enter student name: ";

getline(cin,name);

cout<<"enter number of classes: ";

cin>>nClasses;

cin.ignore();

if(nClasses < 1)

{

cout<<"invalid number of classes";

exit(1);

}

//create dynamic array

classList = new string[nClasses];

for(int i=0;i<nClasses;i++)

{

cout<<"enter class name "<<i+1<<": ";

getline(cin,classList[i]);

}

}

void student::ouput(){

cout<<"Student Name: "<<name<<endl;

cout<<"Number of Classes: "<<nClasses<<endl;

for(int i=0;i<nClasses;i++)

cout<<"Class "<<i+1<<": "<<classList[i]<<endl;

}

void student::reset(){

//resets only if there is something in array

if(classList!=NULL)

{

delete [] classList;

classList = NULL;

}

nClasses=0;

}

int main(){

student s;

s.input();

s.ouput();

s.input();

s.ouput();

return 0;

}