CSci 1113
Final

Name: _____________________________________________

Student ID: _____________________________

Instructions: Please pick and answer any 10 of the 12 problems for a total of 100 points. If you answer more than 10 problems, only the first 10 will be graded. The time limit is 120 minutes. Please write your answers in the space provided. The exam is open book and notes. You may use electronic devices to ONLY look at either an e-book version or electronic notes. You may not use the internet, compiler or any other outside resources. (If you are typing on your keyboard/input device for anything other than ctrl-F to find words in the e-book or notes, this is probably not acceptable.)

Problem (1) [10 points] Write (in C++) a midpoint() member function for the class below (i.e. inside the class). The midpoint can be found by averaging (i.e. mean) the start and end values. Simply write the definition of midpoint(), you do not need to declare it.

class Problem1 {
private:
    double xStart;
    double xEnd;
public:
    Problem1();
};

double Problem1::midpoint()
{
    return (xStart + xEnd) / 2;
}
Problem (2) [10 points] Write (in C++) both a deconstructor and a non-default constructor for the following class. The non-default constructor should take in a single integer as an input and make an array of that length filled with 0’s.

class Problem2 {
private:
    int* vector;
public:
    Problem2();
};

Problem2::Problem2(int len)
{
    vector = new int[len];
    for(int i=0; i < len; i++)
    {
        vector[i] = 0;
    }
}

Problem2::~Problem2()
{
    delete[] vector;
}
**Problem (3)** [10 points] Make a child class for Problem3 below, and override the `sayHello()` function. Then modify main to make it display this new `sayHello()` function you created **while keeping x as a Problem3**. You cannot declare/use any variables other than x.

class Problem3 {
    public:
        void sayHello();
    
    void Problem3::sayHello()
    {
        cout << "Take a hike!" << endl;
    
    int main()
    {
        Problem3* x = new Problem3();
        x->sayHello();
        delete x;
    
    class Problem3Child : public Problem3
    {
    public:
        void sayHello();
    
    void Problem3Child::sayHello()
    {
        cout << "No wai!" << endl;
    
    //Better way to solve fixing:
    class Problem3 {
    public:
        virtual void sayHello(); // changed
    
    int main()
    {
Problem3* x = new Problem3Child(); // changed
x->sayHello();
delete x;
}

// worse way to solve:
/*
int main()
{
    Problem3* x = new Problem3Child(); // changed
    static_cast<Problem3Child*>(x)->sayHello(); // changed
    delete x;
}*/
**Problem (4)** [10 points] Write (in C++) a `copyArray()` function, which returns a dynamically created array identical to the (non-dynamic) array passed in as input. You may assume “`SIZE`” is a global variable of value 5.

```cpp
int original[SIZE] = {1, 5, 3, 6, 3};
int* copy = copyArray(original, SIZE);

for(int i=0; i < SIZE; i++)
{
    if(copy[i] != original[i])
    {
        cout << "You will lose points.\n";
    }
}

int* copyArray(int orig[], int sz)
{
    int* cpy = new int[sz];
    for(int i=0; i < sz; i++)
    {
        cpy[i] = orig[i];
    }
    return cpy;
}
```
Problem (5) [10 points] Write (in C++) an overload function for the “+” operator for the following “Money” class to add together the amounts. You should never have “cents” over 100. This time you do need to declare the function in the class (you can draw an arrow from some text into the class given below) along with the definition.

```cpp
class Money {
private:
    int dollars;
    int cents;
public:
    Money();
    double getValue(); // 10 dollars and 75 cents, this would return 10.75
};

//member-function solution: add to class this line (in public):
Money operator+(Money RHS);

//below class definition:
Money Money::operator+(Money RHS)
{
    Money result;
    result.cents = (cents + RHS.cents) % 100;
    result.dollars = dollars + RHS.dollars + (cents + RHS.cents)/100;
    return result;
}

//friend function solution: add to class this line (in public):
friend Money operator+(Money LHS, Money RHS);

//below class definition:
Money operator+(Money LHS, Money RHS)
{
    Money result;
    result.cents = (LHS.cents + RHS.cents) % 100;
    result.dollars = LHS.dollars + RHS.dollars + (LHS.cents + RHS.cents)/100;
    return result;
}
```
**Problem (6) [10 points]** Write (in C++) an average() function that takes as input an array of Money (as defined in problem 5) and an int denoting the size of the array. This function should correctly find and return the average amount of money across all values in the array passed in. You may assume “SIZE” is a global variable which has value 10. An example code of its use is given below:

```cpp
Money showMeThe[SIZE];
// magically initialize array
double mean = average(showMeThe, SIZE);

double average(Money cash[], int sz)
{
    double sum = 0;
    for(int i=0; i < sz; i++)
    {
        sum += cash[i].getValue();
    }

    return sum/sz;
}
```
Problem (7) [10 points] Write (in C++) the givePointer() function whose use is shown below. You should give as input to this function one int and simply give back a pointer to that int. Will you need to use delete on this? Explain why or why not.

```cpp
int x = 7;
int* xPtr = givePointer(x);
```

```cpp
int* givePointer(int& x)
{
    //NEED CALL BY REF
    return &x;
}
```

No, you would not need to delete this as there is no "new". In fact, if you deleted it, you would crash your program.
Problem (8) [10 points] Find 3 possible places for errors in the following code. Assume this is all the code except for namespaces and includes. Explain specifically what causes each error and whether it is a syntax or logic error:

class Problem8 {
private:
    double y;
};

class Problem8child {
public:
    getY();
};
double Problem8child::getY()
{
    return y;
}

int main()
{
    Problem8child x;
    cout << getY();
    return 0;
}

Logic, not actually a child class.
(you can argue above was a syntax error as getY has no "y" to return.)
Line:
    class Problem8child {

Syntax, no return type;
Line:
    getY();

Syntax, getY() cannot return y, as it is private.
Lines:
private:
    double y;
Syntax, getY() is inside Problem8Child objects (member function)

Line:
    cout << getY();
Problem (9) [10 points] Write (in C++) a recursive function flip() that will cout all the possible coin heads/tails combinations of flipping a coin “n” times (the first argument is a string (hint... hint...) and the second input is the int n). [Hint: use “+=” operator with strings to add letters to an existing string]

```cpp
int main()
{
    flip("", 3);
    // above should cout (in any order, each on a separate line):
    // HHH, HHT, HTH, THH, TTH, THT, HTT, TTT
}

void flip(string s, int times)
{
    if(times == 0)
    {
        cout << s << endl;
        return;
    }

    flip(s+"H", times-1);
    flip(s+"T", times-1);
}
```
Problem (10) [10 points] Assume the file “numbers.txt” has only integers in it. Write a segment of code in C++ (i.e. pretend you are writing somewhere in main) that finds the average of all numbers in this file. You must check for errors when trying to open the file. You may assume all includes have been done for you.

```cpp
ifstream in;
in.open("numbers.txt");

if(in.fail())
{
    cout << "Error";
}
else
{
    double sum = 0;
    int count = 0;
    double num;
    in >> num;
    while(!in.eof())
    {
        count++;
        sum += num;
        in >> num;
    }
    cout << sum/count << endl;
in.close();
}
```
Problem (11) [10 points] Write a short paragraph describing the differences between the functions foo() and bar() provided below. If the input variable “x” was a user-defined class (e.g. Point, Complex, Ship, Money, etc.) instead of an int, what constructors (if any) would be run for each function? (Ignore the fact that “x = 2” might make no sense for a user-defined class.)

```c
void foo(int x)
{
    x=2;
}

void bar(int &x)
{
    x=2;
}
```

foo() copies the value of x, thus there are two boxes, both with the value. When x=2 happens, the variable passed in will not change.

bar() links to the original variable, thus there is really only one box with a value. When x=2 happens, the original variable is changed (i.e. the one passed in).

If the type were a class rather than int...
foo() would run a copy constructor (needs a new box)
bar() would not run any constructors (still only one box)
Problem (12) [10 points] Write a loop using the variable “i” that loops over the range of i values shown (you may use any additional variables if you want). An example is provided below:

(Example) int i: 0, 1, 2, 3, 4
Answer:
for(int i=0; i < 5; i++)
  or
int i=0;
int dummy = 1;
while(i < 5)
  {  
i=i+dummy;
  }

(a) int i: 1, 10
(b) int i: 5, 6, 7, 8, 9, 10
(c) int i: 2, 4, 8, 16, 32
(d) int i: 1234, 123, 12, 1
(e) int i: 1, 1, 2, 3, 5, 8, 13, 21

(a) for(int i=1; i < 11; i+=9)
(b) for(int i=5; i < 11; i++)
(c) for(int i=2; i < 33; i*=2)
(d) for(int i=1234; i > 0; i/=10)
(e)
  int i=1;
  int m1 = 1;
  int m2 = 0;
  while(i < 22)
    {  
i=m1+m2;
     m2 = m1;
     m1 = i;
    }