

**Final Exam, 12/7/08
PIC 10A, Fall 2008**

KEY

Check your TA's name:

_____ **Rachel**

_____ **Aleka**

_____ **Paul**

_____ **Ricardo**

You have 3 hours to complete this exam. You are not allowed to use any books, notes, calculators, or electronic devices. Write your answers carefully. Incomplete, unintelligible, or illegible answers will receive little or no credit. When you are asked to write a program, it is not necessary to comment your code but you are expected to indent appropriately to make your code easier to follow. There are a total of 100 points on this exam.

PAGE	SCORE	POSSIBLE
1		10
2		10
3		12
4		7
5		14
6		12
7		15
8-9		20
TOTAL		100

1.) [2 points] Suppose we have a pointer p. Briefly explain the difference between the two cout statements below.

```
cout << p;
```

```
cout << *p;
```

cout << p; prints the address of the block the pointer p is pointing to (the pointee)

cout << *p; prints the value in the block the pointer p is pointing to (dereferencing)

2.) [2 points] C++ sorts strings *lexicographically*. Explain what lexicographic order means. Then give a specific example that shows lexicographic order is different from alphabetical order.

Lexicographic order sorts strings first by number, then with capital letters, then by lower-case letters. For example, "ZEBRA" comes before "aardvark".

3.) [3 points] Write a single cout statement that prints the following text exactly as below, including the quotation marks.

```
The file is in:
"C:\My Documents"
```

Need escape characters for the line break, quotes, and the backslash.

```
cout << "The file is in:\n\"C:\\My Documents\\\"";
```

4.) [3 points] A cool fact which did not come in lecture is that you can cast between chars and ints. For example the char 'A' has int value 65, 'B' corresponds to 66, 'C' corresponds to 67, and so on in sequential order.

```
cout << (int) 'A';    //Prints the number 65.
cout << (char) 68;    //Prints the letter D.
```

Using this new information, write a simple for loop that prints the 26 letters A through Z.

```
for (int i=0; i<26; i++)
    cout << (char) (i+65);
```

5.) [6 points] Correct the error(s) in each block of code below. You will get points off for changing code that was not in error.

a.)

```
cout << "Enter your first name: ";
string name;
cin << name;
if (name < "Frodo")
    cout << "Hello!";
```

Arrows go the wrong way on input pull.

`cin >> name;`

b.)

```
int* p;
*p = 42;
int* q = p;
*q = 20;
```

Tries to put value 42 into dangling pointer. First need a block.

`int* p = new int;`

c.)

```
int x;
do {
    x = rand();
    cout << "x is between 10 and 100";
} while (x>=10 || x<=100);
```

Creates infinite loop because every number is ≥ 10 OR ≤ 100

`while (x>=10 && x<=100)`

6.) [4 points] The function below is supposed to generate a random number in the range [10,100]. The function compiles correctly, but it contains two serious logic errors.

Explain what the errors are and how to fix them.

```
int generateNumber() {
    srand( (int) time(0));
    return rand()%100;
}
```

i.) We should not seed inside the function. If this function is called twice in one second, it will produce the same number. Seed the program only once.

ii.) Returns random numbers in range 0-99. Use

`return 10+rand()%91;`

7.) [12 points] Write the output of each of the code blocks below.

a.)

```
int d = 31;
do {
    d = d/3;
    if (d%2 == 1)
        d += 4;
    cout << d << "+";
} while (d>2);
```

10+7+2+

b.)

```
string Q = "ABCDEFGH";
cout << Q + Q.substr(2,4) << "\n";
Q.erase(3,2);
cout << Q << "\n";
Q.insert(2, "123");
cout << Q;
```

ABCDEFGHGCDEF

ABCFG

AB123CFG

c.)

```
int a[5] = {10,20,30,40,50};
int* ptr = &a[3];
cout << *(ptr-2) << "\n";
cout << (*ptr)-2 << "\n";
```

20

38

d.)

```
string s[4] = {"Red", "Green", "Blue", "Purple"};
string* p = s;
for (int i=0; i<3; i++)
    cout << (*p) + *(p+i) << "\n";
```

RedRed

RedGreen

RedBlue

8.) [7 points] Write the output of the program below. Track your variables in a walkthrough table as we did in lecture.

```
#include <iostream>
using namespace std;

void fun(int* &c, int* d) {
    *c = 16;
    c = c+2;
    *d = (*c) % 3;
    d = c;
    *d = 2*(*d-1);
    return;
}

int main() {
    int a[5] = {5, 10, 15, 20, 25};
    int* p = a;
    int* q = &a[4];
    fun(p, q);
    for (int i=0; i<5; i++)
        cout << a[i] << " ";
    cout << "\n" << *p;
    cout << "\n" << *q;
    return 0;
}
```

```
16 10 20 20 0
20
0
```

9.) [7 points] The Euclidean norm of a n-dimensional vector $\mathbf{x}$ is defined as

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}.$$

Write a function that takes a vector of doubles as input and returns the norm. You may assume the `<cmath>` library is available. An example is shown below.

$\mathbf{a} =$	1	2	3	4
$Norm = \sqrt{1^2 + 2^2 + 3^2 + 4^2} = \sqrt{30} \approx 5.47722$				

```
double norm (const vector<double>& a) {
    double sum = 0.0;
    for (int i=0; i<a.size(); i++)
        sum += a[i]*a[i];    //Or use pow(a[i],2)
    return sqrt(sum);
}
```

10.) [7 points] The dot product of two n-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$\mathbf{a} \cdot \mathbf{b} = a_1b_1 + a_2b_2 + a_3b_3 + \cdots + a_nb_n.$$

Write a function that takes two vectors of doubles as input and returns the dot product. You may assume the input vectors have the same length. An example is shown below.

$\mathbf{a} =$	1	2	3	4	$\mathbf{b} =$	-6	5	0	8
Dot Product = $1*(-6) + 2*5 + 3*0 + 4*8 = 36$									

```
double dotProduct (const vector<double>&a,
                   const vector<double>& b) {
    double sum = 0.0;
    for (int i=0; i<a.size(); i++)
        sum += a[i]*b[i];
    return sum;
}
```

11.) [7 points] A prime number is a number that is only divisible by itself and 1. Write a boolean function `isPrime` that takes an integer as input and returns true if the number is prime. You may assume the input is a positive number larger than one.

```
bool isPrime (int N) {  
    for (int i=2; i<N; i++)  
        if (N%i == 0)  
            return false;  
    return true;  
}
```

12.) [5 points] A *Mersenne prime* is a prime number of the form $2^x - 1$. Note that not all values of x give a prime number.

$x=4$	$2^4-1=15$	is not prime
$x=5$	$2^5-1=31$	is prime

Write a function which returns true if a given value $x > 1$ generates a Mersenne prime. You may use your function from #11 and the `<cmath>` library.

```
bool isMersennePrime (int x) {  
    return isPrime( pow(2,x)-1 );  
}
```

13.) [15 points] Write a full program (starting from `#include`) that gets the name of a text file from the user and then reports the longest word in the file. In the case when there is a tie for the longest word, report all words with the maximum length in the order in which they appeared in the text file. (*Hint: Use a vector to store the longest words.*) You may assume the file contains only text, with no punctuation. An example output is shown below for the text file "sample.txt".

sample.txt

This is a *resplendent* sample text file to test your *programming* skills
There are three words in this file that are eleven characters long
Your job is to find them

These old school test questions make me *sentimental*

Enter the name of a file: *sample.txt*

The longest words in the file are: *resplendent programming sentimental*

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;

int main() {
    string filename;
    cout << "Enter the name of a file: ";
    cin >> filename;
    ifstream fin;
    fin.open(filename.c_str());

    string word;
    int maxLength = 0;
    vector<string> longestWords;
    while (fin >> word) {
        if (word.length() > maxLength) {
            maxLength = word.length();
            longestWords.resize(0);
            longestWords.push_back(word);
        }
        else if (word.length() == maxLength)
            longestWords.push_back(word);
    }

    cout << "The longest words in the file are: ";
    for (int i=0; i<longestWords.size(); i++)
        cout << longestWords[i] << " ";

    return 0;
}
```

14.) [20 points] The homework grades for students in PIC 10A are stored in a Homework class that keeps track of the student's name and all their homework scores.

- The class constructor takes the name of a student as input.
- The add function adds a homework score to the student's list of scores. You may assume the homework scores are all integers.
- The getName function should return the student's name.
- The getAverage function returns the student's average. Because the professor is not a total jerk, the lowest homework score is dropped when the homework average is computed. You may assume the student has at least 2 homework scores when the average function is called. (*Dropping the lowest score when there is only one score doesn't really make sense.*)
- When two Homework objects are compared, they are compared based on their homework averages.

Write the declaration and function definitions for the Homework class. Some sample code using the Homework class is shown below. You will have to figure out what functions to add based on their usage below.

```
Homework frodosHW ("Frodo Baggins");
frodosHW.add(15);
frodosHW.add(14);
frodosHW.add(3);
cout << "The average after dropping the lowest score for "
      << frodosHW.getName() << " is "
      << frodosHW.getAverage();
/* Note Frodo's average is 14.5, since the 3 is not counted. */

Homework gandalfsHW ("Gandalf the Gray");
gandalfsHW.add(18);
gandalfsHW.add(20);
/* Note Gandalf's average is 20.0, since the 18 is not counted. */

if (frodosHW < gandalfsHW)
    cout << "\nFrodo has a lower homework average.\n"
```

```
class Homework {
public:
    Homework(string new_name);
    void add(int new_score);
    string getName() const;
    double getAverage() const;
    bool operator< (Homework h2) const;
private:
    string name;
    vector<int> scores;
};
```

```

Homework::Homework(string new_name) {
    name = new_name;
}

void Homework::add(int new_score) {
    scores.push_back(new_score);
}

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

//There are several ways to write this function.
//I used one for loop that adds up all the scores.
//Then I subtract out the lowest score at the end.
double Homework::getAverage() const {
    int sum = 0;
    int lowest = scores[0];
    for (int i=0; i<scores.size(); i++) {
        sum += scores[i];
        if (scores[i] < lowest)
            lowest = scores[i];
    }
    sum -= lowest;
    return (double) sum / (scores.size()-1);
}

bool Homework::operator< (Homework h2) const {
    return getAverage() < h2.getAverage();
}

```