

**Final Exam, 3/18/07
PIC 10A, Winter 2007**

KEY

ID NUMBER: _____

Check your TA's name:

_____ **Chris**

_____ **Jukka**

_____ **Stephen**

_____ **Joseph**

_____ **Kree**

_____ **Kwai Tsz (Alvin)**

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 200 points on this exam.

PAGE	SCORE	POSSIBLE
1		20
2		23
3		12
4		18
5		20
6		30
7		22
8		25
9		30
TOTAL		200

1.) [20 points] The following segment of code counts the number of words input by the user.

```
string word;
vector<string> list;
while ( cin >> word )
    list.push_back(word);
cout << "The # of words is " << list.size( );
```

a.) What C++ libraries does this code require?

`<string>` `<vector>` `<iostream>`

b.) Explain the run-time error in this code. (Hint: How many iterations will the loop perform?)

This code gets caught in an infinite loop, because anything the user types will be read in as a string.

c.) Modify the code to stop when the user enters a word with 5 characters like "frodo". You should only change the while boolean condition.

```
while (cin >> word && word.length( ) != 5)
```

d.) Note that if the user entered 1,000,000 words, then the program would store 1,000,000 strings. Modify the code above so that only 1 string needs to be stored.

```
string word;
int numWords = 0;
while (cin >> word)
    numWords++;
cout << "The # of words is " << numWords;
```

2.) [6 points] What is the difference between the `<iostream>` and `<fstream>` libraries? How are they similar?

The `<iostream>` library is for input/output to the console, `<fstream>` is for file I/O. With an `ifstream` `fin` and `ofstream` `fout`, the I/O operations look identical.

`<iostream>` Input: `cin >> x;` Output: `cout << x;`
`<fstream>` Input: `fin >> x;` Output: `fout << x;`

Also both inputs return booleans if the variable `x` was read successfully.

3.) [5 points] In HW7 and HW8, you wrote a `Card` class whose constructor created a random card. Explain why it is important not to seed the program with the `srand` function in the `Card` constructor function.

Seeding in the constructor would start reading the list of random numbers at the specified position every time the `Card` constructor is called. So if 2 cards were created in less than one second, the cards would be identical.

4.) [12 points] Write a for loop that prints the given list of numbers, separated by one space. You may use the `<cmath>` library. The first one is done for you as an example.

1 2 3 4 <i>First four numbers.</i>	<pre>for (int i=1; i <= 4; i++) cout << i << " ";</pre>
-2 -3 -4 -5 -6 <i>Counting down from -2 to -6</i>	<pre>for (int i=-2; i>=-6; i--) cout << i << " ";</pre>
2 4 8 16 32 64 <i>Repeatedly doubles.</i>	<pre>for (int i=1; i<=6; i++) cout << pow(2,i) << " ";</pre>
2 4 6 8 ... 998 1000 <i>Even #s 2 through 1000. Don't print "..."</i>	<pre>for (int i=2; i<=1000; i=i+2) cout << i << " "; OR for (int i=1; i<=500; i++) cout << 2*i << " ";</pre>

5.) [12 points] Each of the code segments below contains an error. Correct the error. You will get points off for changing code that was not in error.

a.)

```
int x;
cout << "Enter a number: ";
cin << x;
cout << "Your number is " << x;
```

Arrows go wrong way: `cin >> x;`

b.)

```
int size = 20;
char letters [size];
for (int i=0; i<size; i++)
    letters[i] = 'J';
```

Array size must be const: `const int size = 20;`

c.)

```
string s = "Frodo";
string* p = s;
cout << "Hello " + *p;
```

Pointer p needs address, not string value: `string* p = &s;`

d.)

```
string s;
getline (cin,s);
switch (s) {
    case "hello":
        cout << "hi";
        break;
    default:
        cout << "bye";
        break;
}
```

Cannot switch on strings. Have to use if-else statement.

```
if (s == "hello")
    cout << "hi";
else
    cout << "bye;
```

6.) [18 points] Write the output of the following segments of code.

a.)

```
int x = 1;
do {
    if (x%3 != 0)
        x += 2;
    else
        x--;
    cout << x << "+";
} while (x < 7 && x >= 2);
```

3+2+4+6+5+7+

b.)

```
int a[4];
a[0] = 2;
for (int i = 1; i < 4; i++)
    a[i] = i + a[i-1];
for (int i = 0; i < 4; i++)
    cout << a[i] << "  ";
```

2 3 5 8

c.)

```
for (int i=1; i<4; i++) {
    for (int j=1; j<=i; j++)
        cout << j << "  ";
    cout << "\n";
}
```

1
1 2
1 2 3

7.) [20 points] Write the output of the following program. Track your variables in a walkthrough table as we did in class.

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

int fun ( int*& p, int* q) {
    *p = 12;
    p = p+3;
    q = p-1;
    *q = *(p+1) + *q;
    *p += 2;
    cout << "p=" << *p << "    q=" << *q << "\n";
    return (*q)%(*p);
}

int main( ) {
    int y[5] = {1,2,3,4,5};
    int* p = y;
    int *q = new int;
    *q = fun ( p, q );
    for (int i=0; i<5; i++)
        cout << y[i] << "    ";
    cout << "\np=" << *p << "    q=" << *q;
    return 0;
}
```

```
p=6 q=8
12 2 8 6 5
p=6 q=2
```

8.) [15 points] Write a function `randomHobbit` which returns the name of one of the four hobbits listed below at random:

Frodo Sam Pippin Merry

All four names should be equally likely to be returned. To receive full credit, you must use a switch statement. An example call to the function is given below.

```
srand ( (int) time(0) );
cout << "A random hobbit is " << randomHobbit( ) << ".\n";
```

```
string randomHobbit( ) {
    int r = 1+rand()%4;
    switch (r) {
        case 1: return "Frodo"; break;
        case 2: return "Sam"; break;
        case 3: return "Pippin"; break;
        case 4: return "Merry"; break;
    }
}
```

9.) [15 points] The Euclidean distance d between two N -dimensional vectors x and y is given by

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_N - y_N)^2}.$$

Note that for $N=2$ dimensions, this is the distance formula we used for the map program in HW5. Write a function `findDistance` that calculates the Euclidean distance between two N -dimensional vectors, $N \geq 1$. You may use the functions in the `<cmath>` library. The prototype is given below.

```
double findDistance (vector<double> x, vector<double> y) {
    double d = 0;
    for (int i=0; i < x.size(); i++)
        d += pow( x[i]-y[i], 2);
    return sqrt(d);
}
```

10.) [22 points] Write a function `printTriangle` that prints out the letters of a given string in a right triangle, as shown below. The first letter of the word appears once in the first row, the second letter appears twice in the second row, and so on. Each column of the triangle should have width 5. You may use the `<iomanip>` library. An example call to the function is shown below, with the corresponding output at right.

<pre>string name = "FRODO"; printTriangle (name);</pre>	F				
	R	R			
	O	O	O		
	D	D	D	D	
	O	O	O	O	O

```
void printTriangle (string s) {
    for (int i=0; i < s.length(); i++) {
        for (int j=0; j <= i; j++)
            cout << setw(5) << s[i];
        cout << "\n";
    }
    return;
}
```

11.) [25 points] Write a full program, starting from `#include`, that gets the name of a text file from the user. The program then reports the longest and shortest word in the file. In the case when there is a tie for the shortest/longest word, report the word that occurs first. You may assume that punctuation contributes to the length of the word, so that "hi!!!!!" is a longer word than "hello". Your output should show double quotes " " around the word. Your answer should not require arrays or vectors. A sample run is shown below for the given input file `input.txt` at right.

Enter file name: <code>input.txt</code>	input.txt
The longest word is "elephants!!!".	Hello Frodo.
The shortest word is "out".	Watch out for those elephants!!!

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

int main ( ) {
    string name, word;
    cout << "Enter file name: ";
    cin >> name;
    ifstream fin;
    fin.open( name.c_str() );
    fin >> word;
    string min = word;
    string max = word;
    while (fin >> word) {
        if ( word.length() < min.length() )
            min = word;
        if ( word.length() > max.length() )
            max = word;
    }
    fin.close();
    cout << "The longest word is \"" << max << "\".\n";
    cout << "The shortest word is \"" << min << "\".\n";
    return 0;
}
```

12.) [30 points] Write a Money class that keeps track of an amount of money. The constructor should take a starting amount as input. The accessors should break the amount down into the proper currency (dollars, quarters, dimes, nickels, pennies). Write the class declaration first and then define the member functions for the class. You'll have to figure out what functions to add based on the sample usage below.

```
Money samsMoney(102.37);    //Sam starts with $102.37.
//Now break down the money into dollars and cents.
cout << "Sam\'s money breaks down as "
      << samsMoney.get_dollars( ) << " dollars, "
      << samsMoney.get_quarters( ) << " quarters, "
      << samsMoney.get_dimes( ) << " dimes, "
      << samsMoney.get_nickels( ) << " nickels, and "
      << samsMoney.get_pennies( ) << " pennies.";
```

Output:

Sam's money breaks down as 102 dollars, 1 quarters, 1 dimes, 0 nickels, and 2 pennies.

Hint: I suggest storing dollars, quarters, dimes, nickels, pennies as private variables.

```
class Money {
public:
    Money (double amount);
    int get_dollars() const;
    int get_quarters() const;
    int get_dimes() const;
    int get_nickels() const;
    int get_pennies() const;
private:
    int dollars, quarters, dimes, nickels, pennies;
};

Money::Money (double amount) {
    dollars = (int) amount;
    int cents = 100*amount-100*dollars;
    quarters = cents/25;
    dimes = (cents-25*quarters)/10;
    nickels = (cents-25*quarters-10*dimes)/5;
    pennies = cents-25*quarters-10*dimes-5*nickels;
}

int Money::get_dollars() const {
    return dollars;
}

int Money::get_quarters() const {
    return quarters;
}
```

```
int Money::get_dimes() const {  
    return dimes;  
}  
  
int Money::get_nickels() const {  
    return nickels;  
}  
  
int Money::get_pennies() const {  
    return pennies;  
}
```

Interesting Note: If you write $\text{cents} = 100 * (\text{amount-dollars})$ in the constructor, you get a crazy round-off error for \$22.81. Of course, we wouldn't take points off for this.