

CS 010 – Intro to Computer Science

Fall - 2003

Mid-term exam – Thursday 10/23 – total 100 points

Time: 70 minutes

You may have on your desks ONLY this exam, a writing implement and an eraser.

You may use a separate sheet(s) of scratch paper. Make good use of this – don't write your answers in the assigned space until you are sure of them!

WRITE CLEARLY IN THE SPACE PROVIDED: illegible responses will not be graded

You will be required to show your student ID when you hand in your exam.

Be sure to read each problem carefully and follow the directions.

Problem 1	14	
Problem 2	8	
Problem 3	5	
Problem 4	5	
Problem 5	5	
Problem 6	15	
Problem 7	10	
Problem 8	10	
Problem 9	8	
Problem 10	20	
TOTAL	100	

1. (14 pts - 1 each) True/False. Please write out "True" and "False". Do not just write T or F.

- a. The compiler will catch all your programming mistakes. *False*
- b. Loops are used when we need our program to make a choice between two or more things. *False*
- c. The body of a do-while loop always executes at least once. *True*
- d. After the following two statements, x will hold the value 4.5: *False*

```
double x;  
x = 9 / 2;
```
- e. Functions may have multiple return statements. *True*
- f. A function may return more than one item. *False*
- g. A good name for a variable representing the cost of an item is x. *False*
- h. The modulus or remainder operator (%) works equally well with integers and doubles. *False*
- i. The C++ language is case-sensitive. *True*
- j. The compiler will catch logic errors. *False*
- k. The expression $!(x < y)$ is equivalent to the expression $(x \geq y)$. *True*
- l. The word `_123` is a valid identifier. *True*
- m. The statement `x++;` is equivalent to the statement `x += 1;` *True*
- n. The input to the compiler is called object code. *False*

2. (8 pts - 2 each) Fill in the blank with the appropriate term.

- a. What is the correct conditional statement to determine if x is between 19 and 99?

$(x > 19 \ \&\& \ x < 99)$

- b. An infinite loop has a Boolean expression that is always *true*.

- c. Write the loop condition to continue a while loop as long as x is negative.

$(x < 0)$

- d. Forgetting to put a semicolon at the end of an assignment statement is an example of a

syntax error.

3. (5 pts) Convert this mathematical expression into a C++ statement. You may assume all the variables have been declared. You may use the following cmath library functions: `double sqrt(double)`
`double pow(double, double)`

$$result = \sqrt{x + \frac{a - b^3}{c}}$$

`result = sqrt(x + (a - pow(b, 3)) / c);`

4. (5 pts) Write a statement showing an example function call given the following function declaration. Pay careful attention to the data types.

```
double my_function(int x, char y);
```

`cout << my_function(3, 'a');`

5. (5 pts) The following function `is_zero` will compile and run, but will give the wrong output. It is supposed to return the bool value `true` when `x` is a 0 and the bool value `false` when `x` is any other number. Right now, it always returns `false`. Circle the error and write the correction to the right.

```
bool is_zero( int x )
{
  if (x = 0)
  {
    return (true);
  }
  else
  {
    return (false);
  }
}
```

`x == 0`

6. (15 pts - 5 each) To the right of each of the following code fragments, show its output.

a.

```
int i = 10;
while (i >= 0)
{
    i -= 3;
    cout << i << " ";
}
cout << i << endl;
```

7 4 1 -2 -2

b.

```
int i = 10;
do
{
    cout << i << " ";
    i -= 3;
} while (i <= 0);
cout << i << endl;
```

10 7

c.

```
int x = 4, y = 3, z = 1;
if (x < y)
{
    z = z + x;
}
else
{
    z = z + y;
}
cout << z << endl;
```

4

7. (10 pts) Write a function declaration and function definition that takes three arguments of type *int* and returns a value of type *double* that is the average of the 3 arguments.

`double avg (int arg1, int arg2, int arg3);`
// returns the average of the 3 parameters

```
double avg(int arg1, int arg2, int arg3)
{
    return ((arg1 + arg2 + arg3) / 3.0);
}
```

8. (10 pts) Write a function that takes as formal parameters number of feet and number of inches. The function should return the total number of inches. For example, if the number of feet is 2 and the number of inches is 9, the function should return the value 33. Provide the function declaration as well as the function definition. Remember to use descriptive names for the function and its formal parameters.

```
int total_inches (int feet, int inches);  
// returns the total number of inches for the given  
// number of feet and inches  
  
int total_inches (int feet, int inches)  
{  
    return ((feet * 12) + inches);  
}
```

9. (8 pts) Fill in the blanks with either Hello or Goodbye so that the second if-else structure would produce the exact same output as the first if-else structure given the same values for x and y.

```
if ((x == 1 && y == 1) || x < y)  
{  
    cout << "Hello";  
}  
else  
{  
    cout << "Goodbye";  
}
```

```
if (x < y)  
{  
    cout << "Hello";  
}  
else  
{  
    if (x == 1)  
    {  
        if (y == 1)  
        {  
            cout << "Hello";  
        }  
        else  
        {  
            cout << "Goodbye";  
        }  
    }  
    else  
    {  
        cout << "Goodbye";  
    }  
}
```

10. (20 pts) Write a program that reads in 10 whole (int) numbers from the user. The program should output the sum of all the positive numbers (greater than 0), the sum of all the negative numbers (less than 0), and the sum of all the numbers whether positive or negative. The user must be allowed to enter the numbers in any order. For example, you can't ask them to enter the negative numbers followed by the positive numbers.

If the user enters 1, -2, 5, 8, -10, 9, -2, -8, 3, 2, the program should output:

```
sum of positive numbers = 28
sum of negative numbers = -22
sum of all numbers = 6
```

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

int main()
{
+5 { int count = 0;
    int number;
    int positive = 0;
    int negative = 0;
    do
    {
+2 { cout << "Enter a whole number: " << endl;
      cin >> number;
      +5 { if (number < 0)
            {
              negative += number;
            }
          else
          {
            positive += number;
          }
        }
      count++;
+2 { } while (count < 10);

+2 cout << "Sum of positive numbers = " << positive << endl;
+2 cout << "Sum of negative numbers = " << negative << endl;
+2 cout << "Sum of all numbers = " << positive + negative << endl;
    }
    return 0;
}
```