CS 7A - Spring 2016 - Midterm 1. 3/17/16

Write all responses on separate paper. Do not use a computer.

1. Write C++ code that declares two variables, `x` and `y`, of type `int`, initializes them to 4 and 7, respectively, then declares a third variable, `sum`, of type `int` and assigns to it the sum of `x+y` and then prints `sum` to the console.

2. Write C++ code that in a loop, prompts the user to `"Enter a string."` and if the string is `"Code in C"`, responds with `"Whispered words of wisdom!"`. Include line feeds (`endl or '\n'`) as needed.

3. What is the output of the following code?

```
int a{3};
int b{8};
b=a++;
cout<<++b;
```

4. What is the output of the following code?

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

int main() {
    int n = 3;
    while (n >= 0)   {
        cout << n * n << endl;
        --n;
    }
    cout << n << endl;
    while (n < 4)
        cout << ++n << endl;
    cout << n << endl;
    while (n >= 0)
        cout << (n /= 2) << endl;
    return 0;
}
```

5. What is the output of the following code?

```
#include <iostream>
int main(){
    int n;
    cout << (n = 4) << endl;
    cout << (n == 4) << endl;
    cout << (n > 3)  << endl;
    cout << (n < 4)  << endl;
    cout << (n = 0)  << endl;
    cout << (n == 0) << endl;
    cout << (n > 0)  << endl;
    cout << (n && 4) << endl;
    cout << (n || 4) << endl;
    cout << (!n)     << endl;
    return 0;
}
```

6. Each of the following programs has error(s). Locate the error(s), classify each error as either a syntax error, a logical error, an overflow error, an underflow error, or a narrowing error.:

(a)
```
int average(int a, int b)
{
    return a + b / 2;
}
```

(b)
```
int main() {
   vector<int> iVector(10);
   myvector[10]=5;
}
```

(c)
```
int  main(void) {
    int x{7777.7};
    char c2{7777};
    cout << c2 << " == " << char(x%256);
}
```

(d)
```
int  foo(int x) {
    return(x+1);
}

int  main() {
    cout << foo(INT_MAX);
}
```

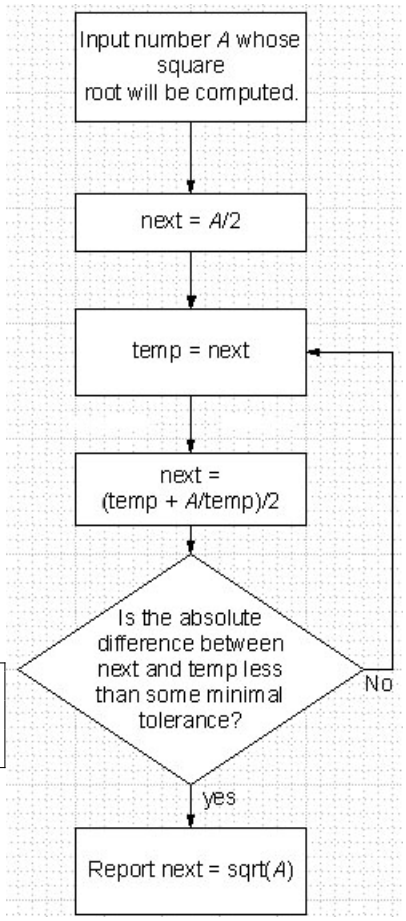
(e)
```
int  main(void) {
    unsigned int a{2};
    unsigned int b{3};
    a -=b;
}
```

7. Explain what the following program does:

```
void foo(int a, int& b) {
  a*=2;
  cout << "\na = " << a;
  b*=2;
}

int main() {
  int x=1, y=3;
  foo(x, y);
  cout << "x=" << x << ", y=" << y;
}
```

8. Write a program that implements the Babylonian algorithm. Use the flow chart at right as a guide, if you like. Remember to use proper syntax and style and to define your variables.



9. Write a C++ program to implement the following pseudo-code for Euclid method of finding the greatest common divisor of `m` and `n`.

```
if m < n, swap(m,n)
while n does not equal 0
   r = m modulo n
   m = n
   n = r
endwhile
output m
```

10. What is the output when the following code fragment is executed?

```
int i = 5, j = 6, k = 7, n = 3;
cout << i + j * k - k % n << endl;
cout << i / n << endl;
```

11. What is the output when the following code fragment is executed?

```
int found = 0, count = 5;
if (!found || --count == 0)
    cout << "danger" << endl;
cout << "count = " << count << endl;
```

12. What is the output when the following code fragment is executed?

```
char ch;
string title = "Titanic";
ch = title[1];
title[3] = ch;
cout << title << endl;
cout << ch << endl;
```

13. The nested conditional statement shown below has been written by an inexperienced C++ programmer. The behavior of the statement is not correctly represented by the formatting.

```
if (n < 10)
    if (n > 0)
        cout << "The number is positive." << endl;
    else    cout << "The number is ______________." << endl;
```

(a) What is the output of the statement if the variable `n` has the value 7 ? If `n` has the value 15 ? If `n` has the value `3` ?

(b) Correct the *syntax* of the statement so that the *logic* of the corrected statement corresponds to the formatting of the original statement. Also, replace the blank with an appropriate word or phrase.

(c) Correct the *formatting style* of the (original) statement so that the new format reflects the logical behavior of the original statement. Also, replace the blank with an appropriate word or phrase.

14. The loop shown below has been written by an inexperienced C++ programmer. The behavior of the loop is not correctly represented by the formatting style.

```
int n = 10;
while (n > 0)
    n /= 2;
    cout << n * n << endl;
```

(a) What is the output of the loop as it is written?

(b) Correct the syntax of the loop so that the logic of the corrected loop corresponds to the formatting of the original loop. What is the output of the corrected loop?

(c) Correct the formatting of the (original) loop so that the new format reflects the logical behavior of the original loop.

15. Remove all the unnecessary tests from the nested conditional statement below.

```
float income;
cout << "Enter your monthly income: ";
cin  >> income;
if (income < 0.0)
    cout << "You are going farther into debt every month." << endl;
else if (income >= 0.0  &&  income < 1200.00)
    cout << "You are living below the poverty line." << endl;
else if (income >= 1200.00  &&  income < 2500.00)
    cout << "You are living in moderate comfort." << endl;
else if (income >= 2500.00)
    cout << "You are well off." << endl;
```

16. Answer the questions below concerning the following fragment of code.

```
int n;
cout << "Enter an integer: ";
cin  >> n;
if (n < 10)
    cout << "less than 10" << endl;
else if (n > 5)
    cout << "greater than 5" << endl;
else    cout << "not interesting" << endl;
```

(a) What will be the output of the fragment above if the interactive user enters the integer value 0 ?

(b) What will be the output of the fragment above if the interactive user enters the integer value 15 ?

(c) What will be the output of the fragment above if the interactive user enters the integer value 7 ?

(d) What values for `n` will cause the output of the fragment above to be "not interesting"?

17. Rewrite the following code fragment so that it uses a "do...while..." loop to accomplish the same task.

```
int n;
cout << "Enter a non-negative integer: ";
cin  >> n;
while (n < 0) {
    cout << "The integer you entered is negative." << endl;
    cout << "Enter a non-negative integer: ";
    cin  >> n;
}
```

18. In the code fragment below, the programmer has almost certainly made an error in the first line of the conditional statement.

(a) What is the output of this code fragment as it is written?

(b) How can it be corrected to do what is the programmer surely intended?

```
int n = 5;
if (n = 0)   // NOTE THE OPERATOR!!!
    cout << "n is zero" << ".\n";
else
    cout << "n is not zero" << ".\n";
cout << "The square of n is " << n * n << ".\n";
```

19. What is the output when the following code fragment is executed?

```
int n, k = 5;
n = (100 % k ? k + 1 : k - 1);
cout << "n = " << n << "    k = " << k << endl;
```

20. What is the output when the following code fragment is executed?

```
int    n;
double x = 3.8;
n = int(x);
cout << "n = " << n << endl;
```

21. What is the output when the following code fragment is executed? Rewrite the fragment to obtain an equivalent code fragment in which the body of the loop is a simple statement instead of a compound statement.

```
int i = 5;
while (i > 0) {
    --i;
    cout << i << endl;
}
```

22. The following loop is an endless loop: when executed it will never terminate. What modification can be made in the code to produce the desired output?

```
cout << "Here's a list of the ASCII values of all the upper"
     << " case letters.\n";
char letter = 'A';
while (letter <= 'Z')
    cout << letter << "  " << int(letter) << endl;
```

23. Write a function named "sum_from_to" that takes two integer arguments, call them "first" and "last", and returns as its value the sum of all the integers between first and last inclusive. Thus, for example,

```
cout << sum_from_to(4,7) << endl;  // will print  22  because 4+5+6+7 = 22
cout << sum_from_to(-3,1) << endl; // will print  5  'cause (-3)+(-2)+(-1)+0+1 = 5
cout << sum_from_to(7,4) << endl;  // will print  22  because 7+6+5+4 = 22 cout << sum_from_to(9,9)
```

24. Write a function named "enough" that takes one integer argument, call it "goal" and returns as its value the smallest positive integer `n` for which $1 + 2 + 3 + \cdots + n$ is at least equal to `goal` . Thus, for example,

```
cout << enough(9) << endl; // will print 4 because 1+2+3+4>9 but 1+2+3<9
cout << enough(21) << endl;// will print 6 because 1+2+...+6>21 but 1+2+...5<21
cout << enough(-7) << endl;// will print 1 because 1> 7 and 1 is the smallest
                           // positive integer
cout << enough(1) << endl; // will print 1 because 1=1 and 1 is the smallest
                           // positive integer
```

DEFINITION: A positive integer `d` is called a divisor of an integer `n` if and only if the remainder after `n` is divided by `d` is zero. In this case we also say that " d divides n ", or that " n is divisible by d ". Here are some examples:
→ 7 is a divisor of 35; that is, 35 is divisible by 7 .
→ 7 is a not a divisor of 27 ; that is, 27 is not divisible by 7. → 1 is a divisor of 19; that is, 19 is divisible by 1 (in fact, 1 is a divisor of every integer n).
→ 12 is a divisor of 0; that is, 0 is divisible by 12.
In C++ one can test the expression `n % d` to determine whether `d` is a divisor of `n`. The greatest common divisor of a pair of integers `m` and `n` (not both zero) is the largest positive integer `d` that is a divisor of both `m` and `n` . We sometimes use the abbreviation "g.c.d." for "greatest common divisor. Here are some examples: 10 is the g.c.d. of 40 and 50; 12 is the g.c.d. of 84 and 132; 1 is the g.c.d. of 256 and 625; 6 is the g.c.d. of 6 and 42 ; 32 is the g.c.d. of 0 and 32.

Write a function named "gcd" that takes two positive integer arguments and returns as its value the greatest common divisor of those two integers. If the function is passed an argument that is not positive (i.e., greater than zero), then the function should return the value 0 as a sentinel value to indicate that an error occurred. Thus, for example,

```
cout << gcd(40,50) << endl;   // will print  10
cout << gcd(256,625) << endl; // will print  1
cout << gcd(42,6) << endl;    // will print  6
cout << gcd(0,32) << endl;    // will print  0  (even though  32  is the gcd.)
cout << gcd(10,-6) << endl;   // will print  0  (even though  2  is the gcd.)
```

25. Write a function named "digit_name" that takes an integer argument in the range from 1 to 9 , inclusive, and prints the English name for that integer on the computer screen. No newline character should be sent to the screen following the digit name. The function should not return a value. The cursor should remain on the same line as the name that has been printed. If the argument is not in the required range, then the function should print "digit error" without the quotation marks but followed by the newline character. Thus, for example, the statement

    ```
    digit_name(7);
    ```

    should print `seven` on the screen; the statement

    ```
    digit_name(0);
    ```

    should print `digit error` on the screen and place the cursor at the beginning of the next line.

26. Write a function named "reduce" that takes two positive integer arguments, call them "num" and "denom", treats them as the numerator and denominator of a fraction, and reduces the fraction. That is to say, each of the two arguments will be modified by dividing it by the greatest common divisor of the two integers. The function should return the value `0` (to indicate failure to reduce) if either of the two arguments is zero or negative, and should return the value `1` otherwise. Thus, for example, if `m` and `n` have been declared to be integer variables in a program, then

    ```
    m = 25;
    n = 15;
    if (reduce(m,n))
        cout << m << '/' << n << endl;
    else
        cout << "fraction error" << endl;
    ```

will produce the following output:

```
5/3
```

Note that the values of `m` and `n` were modified by the function call. Similarly,

```
m = 63;
n = 210;
if (reduce(m,n))
    cout << m << '/' << n << endl;
else
    cout << "fraction error" << endl;
```

will produce the following output:

```
3/10
```

Here is another example.

```
m = 25;
n = 0;
if (reduce(m,n))
    cout << m << '/' << n << endl;
else
    cout << "fraction error" << endl;
```

will produce the following output:

```
fraction error
```

The function `reduce` is allowed to make calls to other functions that you have written.

27. Write a function named "swap_floats" that takes two floating point arguments and interchanges the values that are stored in those arguments. The function should return no value. To take an example, if the following code fragment is executed

```
float x = 5.8, y = 0.9;
swap_floats (x, y);
cout << x << "   " << y << endl;
```

then the output will be

```
0.9   5.8
```

28. Write a function named "sort3" that takes three floating point arguments, call them "x" , "y" , and "z" , and modifies their values, if necessary, in such a way as to make true the following inequalities: `x<y<z`. The function should return no value. To take an example, if the following code fragment is executed

```
float a = 3.2, b = 5.8, c = 0.9;
sort3 (a, b, c);
cout << a << "   " << b << "   " << c << endl;
```

then the output will be

```
0.9   3.2   5.8
```

The function `sort3()` is allowed to make calls to other functions that you have written.

29. Write a function named "`reverse`" that takes as its argument a vector of floating point values;

    The function must reverse the order of the values in the vector. Thus, for example, if the vector that's passed to the function looks like this:

0	1	2	3	4
5.8	2.6	9.0	3.4	7.1

    then when the function returns, the array will have been modified so that it looks like this:

0	1	2	3	4
7.1	3.4	9.0	2.6	5.8

    The function should not return any value.

30. Write a function named "`sum`" that takes as its arguments a vector of floating point values. The function should return as its value the sum of the floating point values in the array. Thus, for example, if the vector that's passed to the function looks like this:

0	1	2	3	4
5.8	2.6	9.0	3.4	7.1

    then the function should return the value 27.9 as its value.

31. Write a function named "`location_of_largest`" that takes as its argument a reference to a vector of integer values. The function should return as its value the index of the cell containing the largest of the values in the array. Thus, for example, if the array that's passed to the function looks like this:

0	1	2	3	4
58	26	90	34	71

    then the function should return the integer 2 as its value. If there is more than one cell containing the largest of the values in the array, then the function should return the smallest of the subscripts of the cells containing the largest values. For example, if the array that's passed to the function is

0	1	2	3	4	5	6
58	26	91	34	70	91	88

    then the largest value occurs in cells 2 and 5, so the function should return the integer value 2 .

32. Consider the complete program below:

```
#include "std_lib_facilities.h"

const int MAX{8};

void cycle_sort(vector<int>&);

int main() {
    vector<int> a(MAX);
    int i;

    cout << "enter the elements into array :";
    for (i = 0;i < MAX; i++)     {
        cin >> a[i];
    }
    cycle_sort(a);
    cout << "sorted elements are :\n";
    for (i = 0;i < MAX; i++)  {
        cout << a[i] << " ";
    }
}

/* sorts elements using cycle sort algorithm */
void cycle_sort(vector<int>& a) {
    int item, pos, i, j, k;

    for (i = 0; i < MAX; i++)     {
        item = a[i];
        pos = i;
        do {
            k = 0;
            for (j = 0; j < MAX; j++) {
                if (pos != j && a[j] < item) {
                    k++;
                }
            }
            if (pos != k)  {
                while (pos != k && item == a[k]) {
                    k++;
                }
```

```
40                  swap(a[k],item);
                    pos = k;
42              }
            } while (pos != i);
44      }
}
```

(a) Which line contains a function declaration/prototype?

(b) Which lines contain the definition of this function?

(c) Carefully describe the parameter list of this function.

(d) What is the condition for continuing the do-loop of the function?

(e) What is the condition for ending the for-loop in the body of this do-loop?

(f) Suppose the user enters `1 3 2 4 5 7 8 6`. Trace the values of the various components of the program as the instructions are followed, step by step:

a	i	item	pos	k	j	`pos!=j && a[j]<item`	`pos!=k`	pos!=i
{1 3 2 4 5 7 8 6}	0	1	0	0	0	`false` && `false`	`false`	`false`