CS 7A - Fall 2016 - Midterm 1 10/20/16

Write responses to questions 1 and 2 on this paper or attach additional sheets, as necessary. For all subsequent problems, use separate paper. Do not use a computer.
Assume the necessary libraries are included and appropriate namespaces are used.

1. Match each of the following terms on the left with its definition on the right.

(a) **declaration 6**
(b) **byte 9**
(c) **narrowing 8**
(d) **invariant 11**
(e) **Compile errors 12**
(f) **definition 3**
(g) **variable 5**
(h) **function 2**
(i) **type 1**
(j) **object 7**
(k) **bit 4**
(l) **logic errors 10**

(1) Something that defines a range of possible values and a set of operations for an object.
(2) A named unit of code that can be invoked (called) from different parts of a program; a logical unit of computation.
(3) A declaration that allocates and memory and assigns all necessary value(s) for it.
(4) the most basic unit of information in a computer that can have the value 0 or 1.
(5) A named object of a given type that contains a value unless uninitialized.
(6) The specification of a name with its type in a program.
(7) Some memory that holds a value of a given type.
(8) A conversion of types that may put a value into an object that is too small to hold it.
(9) The basic unit of addressing in most computers.
(10) Errors found by the programmer looking for the causes of erroneous results.
(11) Something that must be always true at a given point (or points) of a program.
(12) Syntax or type errors.

2. If x contains the value 3 before the following instruction is executed

```
for(int i = 0; i < 4; i+=2)
  x *= 5;
```

(a) What is the value of x after the loop runs?
ANS: $3 \cdot 5^2 = 3 \cdot 25 = 75 = 1001011_2$,

(b) Rewrite the loop as an equivalent `while` loop.

```
int x = 3, i = 0;
while(i<4)
{   x *= 5;
    i += 2;
}
```

(c) Rewrite the loop as an equivalent `do-while` loop.

```
int x = 3, i = 0;
do
{   x *= 5;
    i += 2;
} while(i<4);
```

3. Write 216

   (a) in base 2 form. That is, determine binary digits $a_7, a_6, \ldots a_0$ such that $a_7 \cdot 2^7 + a_6 \cot 2^6 + \cdots a_0 \cdot 2^0 = 216$
   ANS: $216 = 11011000_2$ You can get this just by fiddling with sums of powers of 2 or by repeatedly dividing by 2 and noting the remainder at each stage:
   The sequence of remainders are the digits of the binary representation in increasing significance.

   $216 = 2 \cdot 108 + \mathbf{0}$
   $108 = 2 \cdot 54 + \mathbf{0}$
   $54 = 2 \cdot 27 + \mathbf{0}$
   $27 = 2 \cdot 13 + \mathbf{1}$
   $13 = 2 \cdot 6 + \mathbf{1}$
   $6 = 2 \cdot 3 + \mathbf{0}$
   $3 = 2 \cdot 1 + \mathbf{1}$
   $1 = 2 \cdot 0 + \mathbf{1}$

   (b) in base 16 form. That is, determine hexadecimal digits a_1, a_0 such that $a_1 \cdot 16 + a_0 = 216$
   ANS: $216 = \texttt{D8}_{16}$. You can get this just by fiddling with sums of multiples of powers of 16 or by repeatedly dividing by 16 and noting the remainder at each stage:
   $216 = 16 \cdot 13 + \mathbf{8}$
   $13 = 16 \cdot 0 + \mathbf{13}$
   The sequence of remainders are the digits of the hexadecimal representation in increasing significance. The number 13 is represented by D in hexadecimal.

4. Suppose we declare a variable like so: `string place = "BED";`

   (a) How many bytes of memory would the variable `place` require in memory? Assume that the string is appended by the '\0', or "NULL" character, which takes one byte of all 0 bits.
   ANS: Each character requires one byte, plus the Null character byte for a total of 4 bytes.

   (b) How is `place` represented as a sequence of bits in memory? *Hint:* the ASCII code for `'A'` is 65.
   B= $66 = 01000010_2$, E= $69 = 01000101_2$ and D=$68 = 01000100_2$ and the null character is all zeros, so `place` would be represented by $01000010010001010100010000000000_2$

5. Suppose you have a point on a plane represented by the variables `ptX` and `ptY` (its x and y coordinates). Suppose you also have a rectangle, whose upper left corner is represented by the variables `left` and `top`, and whose bottom right corner is represented by the variables `right` and `bot`. What conditional statement would you write to determine whether or not `(ptX,ptY)` is inside the rectangle or not? That is how would you fill in the condition in the statement below:

```
if(/* fill in condition here */)
    cout << "\nThe point (" << ptX << "," << ptY << ") is inside the rectangle.";
else
    cout << "\nThe point (" << ptX << "," << ptY << ") is not inside the rectangle.";
```

ANS:

```
if(left<ptX && ptX<right && bot<ptY && ptY<top)
```

6. Consider the program shown at right below:

(a) What prototypes need be declared before `main()` ?
ANS:
```
vector<int> bubbleSort(vector<int>);
void printVector(vector<int>);
```

(b) What is the return type of of `bubbleSort()` ?
`bubbleSort()` returns a `vector<int>` type.

(c) What is the value of `a.size()` ?
`a.size()=4`

(d) What is `temp` used for in `bubbleSort()` ?
`temp` is used in the process of swapping the values of `a[i]` and `a[i+1]`

(e) Tabulate values of `k`, `i`, `a[i] > a[i+1]` (`true` or `false`) and `a` (list the entire vector at each iteration) as the body of `bubbleSort()` is executed.

```
1  int main()
   {  vector<int> a{3,2,6,1}, b;
3     printVector(a);
      b = bubbleSort(a);
5     printVector(b);
   }
7  vector<int> bubbleSort(vector<int> a)
   {  int temp;
9     for (int k=0;k<a.size();k++)
      {  for(int i=0;i<a.size()-1;i++)
11       {   if (a[i] > a[i+1])
             {  temp = a[i];
13              a[i] = a[i+1];
                a[i+1] = temp;
15           }
         }
17    }
   }
19 void printVector(vector<int> a)
   {   for(int i=0; i<a.size(); i++)
21         cout<<a[i]<<"␣";
      cout<<endl;
23 }
```

ANS:

k	i	a[i]>a[i+1]	a
0	0	true	{ 3,2,6,1}
0	1	false	{ 2,3,6,1}
0	2	true	{ 2,3,6,1}
1	0	false	{ 2,3,1,6}
1	1	true	{ 2,3,1,6}
1	2	false	{ 2,1,3,6}
2	0	true	{ 2,1,3,6}
2	1	false	{ 1,2,3,6}
2	2	false	{ 1,2,3,6}
3	0	false	{ 1,2,3,6}
3	1	false	{ 1,2,3,6}
3	2	false	{ 1,2,3,6}

This is not, however, a proper bubblesort. To fix the code, change line 10 to

```
for (int i = 0; i < a.size()-1-k; i++)
```

Notice the added "`-k`". This makes the algorithm considerably more efficient:

k	i	a[i]>a[i+1]	a
0	0	true	{ 3,2,6,1}
0	1	false	{ 2,3,6,1}
0	2	true	{ 2,3,6,1}
1	0	false	{ 2,3,1,6}
1	1	true	{ 2,3,1,6}
2	0	true	{ 2,1,3,6}

Then at the last, the first two elements are swapped and the vector is sorted.

(f) There is an error in `bubbleSort()`. How can you fix it? *Hint:* what happens on line 5?
ANS: `bubbleSort()` needs to have a `return a;` statement at the end.

7. Consider the following pseudocode:

(a) Write a C++ program to implement the pseudocode:

```
Set den to one
Get num from user
Set term = num/den (a floating point value)
Set i = 1
set sum = 1
While i less than or equal to three
    i is increased by 1
    term is multiplied by num
    den is multiplied by i
    term is divided by den
    term is added to sum
Print sum
```

```
int main() {
    int den{1}, num;
    double term, sum{1.};
    cout << "\nEnter num: ";
    cin >> num;
    term = (double)num/den;
    int i = 1; //, sum = 1;
    while(i <= 3) {
        ++i;
        term *= num;
        den *= i;
        term /= den;
        sum += term;
    }
    cout << "\nsum = " << sum;
}
```

(b) Tabulate values of i and values of `term`, `den` and `sum` as the loop is executed if the user enters 2:

i	term	den	sum
1	2	1	1
2	4	2	
	2		3
3	4	6	
	$\frac{2}{3}$		$3\frac{2}{3}$
4	$\frac{4}{3}$	24	
	$\frac{1}{18}$		$3\frac{13}{18} = 3.7\overline{2}$

8. Write a function that takes a positive integer and determines whether or not it is divisible by some square number, $k^2 > 1$. That is,
`precondition:` A positive integer.
`postcondition:` `true` if there is some integer $k > 1$ such that n modulo k^2 is zero, otherwise, `false`
ANS:

```
bool squareFree(int n) {
    for(int i = 2; i < sqrt(n); ++i)
        if(n%(i*i)==0) return false;
    return true;
}

int main() {
    int n;
    cout << "\nEnter a positive integer to see if it's square-free: ";
    while(cin>>n)
    {   if(squareFree(n)) cout << n << " is square-free.";
        else cout << n << " is not square-free.";
        cout << "\nEnter a positive integer to see if it's square-free: ";
    }
}
```

9. Consider the code below, and assume all the needed libraries are included and `using namespace std;`

```
1  const int maxint = 25;
   void print(vector<bool> v);
3  int main() {
       vector<bool> primes(maxint);
5      for(int i = 2; i < maxint; ++i)
           primes[i] = 1;
7      for(int i = 2; i < sqrt(maxint); ++i) {
           for(int j = 2; j <= maxint/i; ++j)
9              primes[j*i] = 0; /// multiples of i are composite
       }
11     print(primes);
   }
13 void print(vector<bool> v) {
       int j = 0;
15     for(int i = 0; i < v.size(); ++i) {
           if(v[i]) {
17             cout << i << '\t';
               if((j++ +1)%10==0) cout << endl;
19         }
       }
21 }
```

(a) Describe what the declaration on line 4 does.
ANS: This declares `primes` to be a `vector` of `bool` with size 0.

(b) Describe in detail what the loop on lines 5,6 does.
These lines set the value of `primes[i]` to `true` for `i=2,3,..,24`

(c) Complete the table below which threads the values of quantities as the loop on lines 7-10 is executed (see table at right) Note that `i` would not get to 5 since $5 \not< \sqrt{25}$, but then 25 would not be seen as composite, so the inequality should be changed to "`<=`".

(d) Why is this code for computing prime numbers not as efficient as it could be? What could you do to improve the efficiency? Here's the more efficient code we wrote in class:

```
1  int main()
   {   unsigned maxint = 10000000;
3      vector<bool> primes(maxint);
       for(int i=2; i<=sqrt(maxint); )
5      {  for(unsigned j=i; j<=maxint/i; ++j)
              primes[j*i] = 1; /// composites
7          while(primes[++i]);
       }
9      print(primes);
   }
```

This is more efficient in that `j` starts at `i` instead of 2 and `i` is quickly "scooted" to the next prime by the crafty gadget of `while(primes[++i]);`

(e) Describe the logic of how the `print()` function works.
The print statement traverses the `vector` of `bool` and prints its index only if the value there (`v[i]`) is `true`.

i	j	i*j	primes[i*j]	maxint/i
2	2	4	0	12
	3	6	0	
	4	8	0	
	5	10	0	
	6	12	0	
	7	14	0	
	8	16	0	
	10	20	0	
	12	24	0	
3	2	6	0	8
	3	9	0	
	4	12	0	
	5	15	0	
	6	18	0	
	7	21	0	
	8	24	0	
4	2	8	0	6
	3	12	0	
	4	16	0	
	5	20	0	
	6	24	0	
5	2	10	0	5
	3	15	0	
	4	20	0	
	5	25	0	
⋮	⋮	⋮	⋮	⋮