CS 007A – Midterm 1 Practice – Chapters 1—5

1. Consider the 4 bytes stored in contiguous memory, as shown to the right.
   a. Determine the decimal representation of these 4 one-byte integers:
   b. Determine the decimal form of these, interpreting the data as 2 two-byte integers are stored in these 4 bytes.
   c. Interpret the 4 bytes as 4 ASCII characters. Note that the ASCII value of A is 65.

⋮
01000001
01010011
01001011
01011001
⋮

2. Write 125 in binary form.

3. Write 125 in Hex form.

4. Convert the hex number abba to decimal form.

5. Find as many errors in the following code as you can. Classify each error as a syntax error or a logic error and describe how to fix it so it works properly.

```
//This program uses a loop to raise a number to a power.
#include <iostream>
using namespace std;

int main()
{  int num, bigNum, power, count;
   cout << "Enter an integer: ";
   cin >> num;
   cout << "\nWhat power do you want it raised to? ";
   cin >> power;
   bigNum = num;
   while (count++ < power);
       bigNum *= num;
   cout << "\nThe result is " << bigNum << endl;
      return 0;
   }
```

6. Consider the following C++ program. Is there an error? If so what kind of error? What will be the result of attempting to compile and execute the program?

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

int main()
{  for(int i = 0; i < 2; ++i)
         for(int j = 0; j < 2; ++ j)
         {
               cout << "i = " << i << " ";
               --i;
               cin.get();
         }
   return 0;
}
```

7. Evaluate the following expressions, step by step, assuming integer variables `a = 52`, `b = 13`, `c = 4`, `d = 5` and `float e = 2.71828`
   If the expression doesn't parse, explain why.
   a. `a / d % b * c`
   b. `a % c == b`

c. `!(b + d  != c * c)`
d. `d % b * c > 5 || c % b * d < 7`
e. `(((a + 1)*b + 2) % c + 3)/d + e`

8. What is the output of the following code fragment?

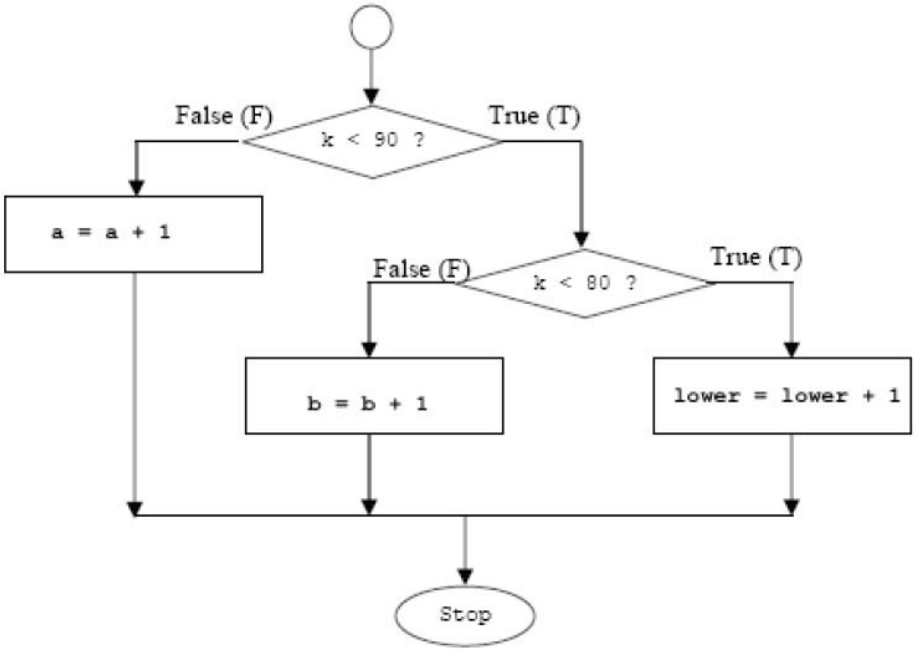
```
for (int ct = 1; ct <= 3; ct++) {
    cout << ct;
    for (int i = 3; i <= 4; i++) {
        cout << ct << " " << i;
    }
    cout << "do re mi";
}
```

9. What is the output produced by the following program if the user enters 4?

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

int main()
{
      setiosflags(ios::right);
      int n;
      cout << "How many columns? (1-16): ";
      cin >> n;
      for (int x=1; x <= n; x++)
      {
            for (int y=1; y <= n; y++)
                  cout << setw(5) << x*y;
            cout << endl;
      }
}
```

10. Write code to implement the following flow chart. The user will enter 10 numbers between 0 and 100 and the program will count how many are ≤ 90, how many are in the interval [80, 90) and how many are less than 80 and then report the results.



11. Write a program to
    a. Find the square root of the sum of the first 100 even numbers, correct to 10 significant digits. You may use not the `cmath` library.

    b. Find the reciprocal of the product of the first 10 odd numbers correct to 10 significant digits.

    c. Fill up an array of 3 elements with 3 random numbers between 1 and 10 and then compute the cube root of their product. Do not use the `cmath` library.

    d. Compute the value of the zeta function, $\zeta(s) = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \cdots$ for any positive integer $s > 1$.

12. Consider the following program fragment.

```
int accumulator = 0, sam, pam;
cout << "\nEnter integers for sam and pam: ";
cin >> sam >> pam;
while (true )
{
   if (pam == 0) break ;
   accumulator += ((pam % 2 == 1) ? sam : 0);
   pam /= 2;
   sam *= 2;
}
cout << accumulator << "\n";
```

    a. Complete the following tables until the program completes, or indicate that it's an infinite loop:

sam	pam	accumulator
5	4	0
⋮	⋮	⋮

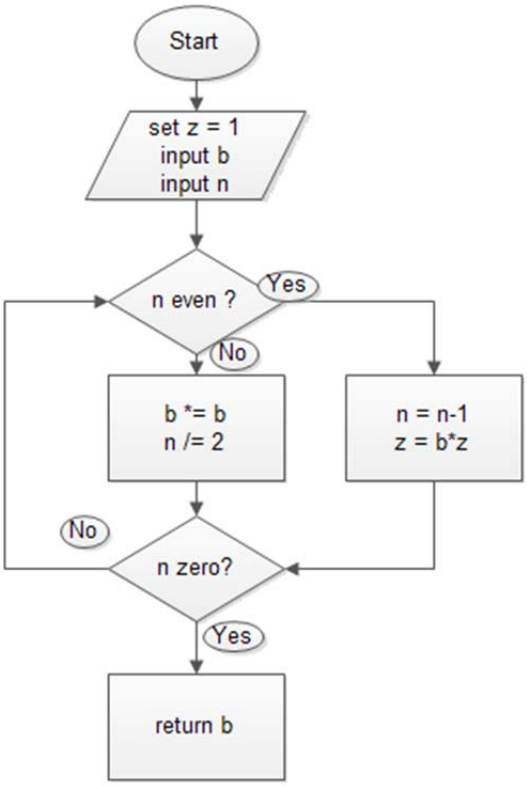
sam	pam	accumulator
6	17	0
⋮	⋮	⋮

    b. In a few words, describe the output of this program in terms of the input values of `sam` and `pam`, and how it works.

13. Suppose that p = 0x0012FF50, &a = 0x0012FF44, &p = 0x0012FF38.
    What will be output by the following code?

```
int main()
{ int a[] = { 22, 33, 44, 55, 66, 77, 88, 99 };
  int* p = &a[3];                          // p points to a[3]
  cout << "p = " << p << ", *p = " << *p;
  cout << "\n&a = " << &a;                 // ok: a is an lvalue
  cout << "\n&p = " << &p;                 // ok: p is an lvalue
  cout << "\n&(a[5]) = " << &(a[5]);      // ok: a[5] is an lvalue
  cout << "\n&(*(a+5)) = " << &(*(a+5));// ok: *(a+5) is an lvalue
  cout << "\n&(*(p+2)) = " << &(*(p+2));// ok: *(p+2) is an lvalue
  cout << "\n&(p+2) = " << p+2; //&(p+2);  // ERROR: p+2 is not an lvalue
  cout << endl;
}
```

14. Write a C++ program to implement the flow chart below.



CS 007A – Midterm 1 Practice Problems Solutions– Chapters 1—5

⋮
01000001
01010011
01001011
01011001
⋮

1. Consider the 4 bytes stored in contiguous memory, as shown to the right.
   a. Determine the decimal representation of these 4 one-byte integers:
      SOLN: $01000001_2 = 2^6+2^0 = 64 + 1 = 65$
      $01010011_2 = 2^6+2^4 + 2^1 + 2^0 = 64 + 16 + 2 + 1 = 83$
      $01001011_2 = 2^6+2^3 + 2^1 + 2^0 = 64 + 8 + 2 + 1 = 75$
      $01011001_2 = 2^6+2^4 + 2^3 + 2^0 = 64 + 16 + 8 + 1 = 89$
   b. Determine the decimal form of these, interpreting the data as 2 two-byte integers are stored in these 4 bytes.
      SOLN: Of course there are two ways to do this: either the most significant byte is first, or the least significant byte is first. That is, either
      $0101001101000001_2 = 83*2^8 + 65 = 21313$
      Or
      $0100000101010011_2 = 65*2^8 + 83 = 16723$
      Similarly for the second pair: it's either
      $0100101101011001_2 = 75* 2^8 + 89 = 19289$
      or
      $0101100101001011_2 = 89* 2^8 + 75 = 22859$
   c. Interpret the 4 bytes as 4 ASCII characters. Note that the ASCII value of A is 65.
      SOLN: ASKY

2. Write 125 in binary form.
   SOLN: $125 = 64 + 32 + 16 + 8 + 4 + 1 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 1111101$

3. Write 125 in Hex
   SOLN: $125 = 16*7 + 13 = 7D$

4. Convert the hex number abba to decimal form.
   SOLN: abba = $10*16^3 + 11*16^2 + 11*16 + 10 = 40960 + 2816 + 176 +10 = 43962$

5. Find as many errors in the following code as you can. Classify each error as a syntax error or a logic error and describe how to fix it so it works properly.

```
//This program uses a loop to raise a number to a power.
#include <iostream>
using namespace std;

int main()
{  int num, bigNum, power, count;
   cout << "Enter an integer: ";
   cin >> num;
   cout << "\nWhat power do you want it raised to? ";
   cin >> power;
   bigNum = num;
   while (count++ < power);
       bigNum *= num;
   cout << "\nThe result is " << bigNum << endl;
      return 0;
}
```

   SOLN: ANS: There is a logic error that looks like a syntax error: the semicolon after `while (count++ < power);`
   and another logic error: `count` is never initialized. If the semicolon is deleted and `count` is initialized to 0, then the algorithm should successfully raise the user supplied `int` to the user supplied positive `int`.

6. Consider the following C++ program. Is there an error? If so what kind of error? What will be the result of attempting to compile and execute the program?

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

int main()
{  for(int i = 0; i < 2; ++i)
         for(int j = 0; j < 2; ++ j)
         {
               cout << "i = " << i << " ";
               --i;
               cin.get();
         }
   return 0;
}
```

ANS: There is a logic error. The output of the program will proceed in an infinite loop like so:

```
i = 0
i = -1
i = -1
i = -2
i = -2
i = -3
i = -3…
```

To see this, note that initially, `i = j = 0` and `"i = 0"` is printed. Then `i` is decremented by 1, so `i = -1`, and the program waits for a keystroke, say, "Enter." Then `j = 1` and `"i = -1"` is printed and `i` is decremented again so `i = -2`, but then we arrive at the update for the outer for loop, which increments `i` again to `i = -1`, so `"i = -1"` is printed again. Now `i` is decremented to -2 again and `j` counts to 1, `"i = -2"` is printed and `i` decremented to `i = -3`. After pressing "Enter" again, control passes back to the outer for loop, which increments `i` again so `i = -2` and the inner for loop prints `"i = -2"`, decrements `i` again (so `i = -3`) and waits for a keystroke. Everytime the user presses a key (in the above example, simply the "Enter" key) the pattern repeats with `i` steadily decreasing by one every time the outer for loop repeats, printing two instances of the same value of `i` and since the sentinel for the outer for loop is `i < 2`, it's an infinite loop.

7. Evaluate the following expressions, step by step, assuming integer variables `a = 52`, `b = 13`, `c = 4`, `d = 5` and `float e = 2.71828`
If the expression doesn't parse, explain why.
   a. `a / d % b * c`
      SOLN: According the precedence of operators in C++ (described, for instance, at http://en.cppreference.com/w/cpp/language/operator_precedence) the operators and all have the same precedence and are associated left to right.
      `a / d % b * c = 10 % b * c = 10 * 4 = 40`
   b. `a % c == b` The comparison operator has lower precedence than the modulus operator, so this is equivalent to `0==13,` which is false, or 0.
   c. `!(b + d  != c * c)`
      The negation operator has high precedence, but the parentheses have higher precedence, so we look inside the parentheses first. The addition and multiplication operators have higher precedence than the comparison operator, so this is simplified to `!(18  != 16)`. Now the comparison inside the parentheses is true, which is negated to false or, `= 0`
   d. `d % b * c > 5 || c % b * d < 7`
      Here the logical "or" has low precedence and we simplify the expressions on the left and right of it first, and since the comparison operators have lower precedence than the arithmetic operators, this goes like so: `5 * 4 > 5 || 4 * 5 < 7` and then

`20 > 5 || 20 < 7` and then `true || false` which is `true` or 1.
`= 20 > 5 || 20 <  7`

e. `(((a + 1)*b + 2) % c + 3)/d + e = 3.71828`
This is largely driven by the parentheses:
`((53*13 + 2) % 4 + 3)/5 + 2.71828`
`((689 + 2) % 4 + 3)/5 + 2.71828`
`(691 % 4 + 3)/5 + 2.71828`
`(3 + 3)/5 + 2.71828`
`6/5 + 2.71828`
`1 + 2.71828 = 3.71828`
Note the promotion of the int on the left to the float type on the last step.

8. What is the output of the following code fragment?

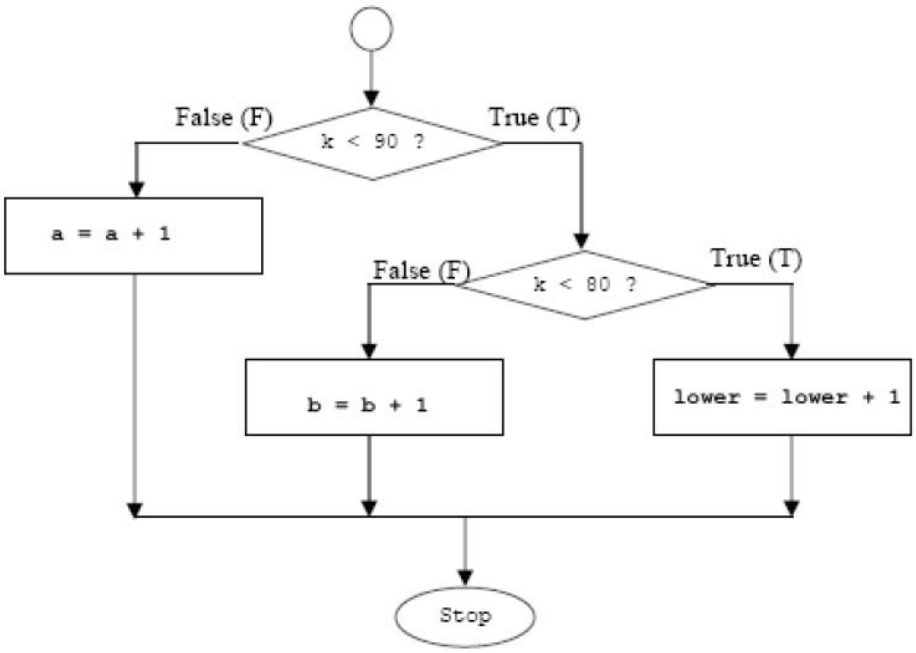
```
for (int ct = 1; ct <= 3; ct++) {
   cout << ct;
   for (int i = 3; i <= 4; i++) {
      cout << ct << " " << i;
   }
   cout << "do re mi";
}
SOLN:
11 31 4do re mi22 32 4do re mi33 33 4do re mi
```

9. What is the output produced by the following program if the user enters 4?

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

int main()
{
      setiosflags(ios::right);
      int n;
      cout << "How many columns? (1-16): ";
      cin >> n;
      for (int x=1; x <= n; x++)
      {
            for (int y=1; y <= n; y++)
                  cout << setw(5) << x*y;
            cout << endl;
      }
}
SOLN:
    1    2    3    4
    2    4    6    8
    3    6    9   12
    4    8   12   16
```

Write code to implement the following flow chart. The user will enter 10 numbers between 0 and 100 and the program will count how many are ≤ 90, how many are in the interval [80, 90) and how many are less than 80 and then report the results.



SOLN:

```
int main() {
    int a = 0, b = 0, lower = 0;
    for(int i = 0; i < 10; ++i)
    {
        cin >> k;
        if(k >= 90) ++a;
        else if(k >= 80) ++b;
        else ++lower;
    }
    cout << "\nThere are " << a
         << " 90 or more.\n" << b
         << " in [80,90) and \n"
         << lower << " less than 80.";
}
```

10. Write a program to
    a. Find the square root of the sum of the first 100 even numbers, correct to 10 significant digits. You may use not the `cmath` library.
    SOLN:

```
#include <iomanip>
#include <iostream>
using namespace std;
double root(double);

int main()
{
      double sum = 0;
      cout << setprecision(10);
      for (double n = 2; n < 100; n += 2)
            sum += n;
      cout << "\nsqrt(0 + 2 + ... + 98) = " << root(sum) << endl;
}

double root(double A)
{     // Use the babylonian algorithm to compute the square root
      double x1 = A, x2 = x1/2;
      const double tolerance = 1e-14;
      while(x2 - x1 > tolerance || x1 - x2 > tolerance)
      {
```

```
            x1 = x2;
            x2 = (x1 + A/x1)/2;
      }
      return x2;
}
```

Output:
sqrt(0 + 2 + ... + 98) = 49.49747468

b. Find the reciprocal of the product of the first 10 odd numbers correct to 10 significant digits.
SOLN:

```
int main()
{
      double product = 1;
      cout << setprecision(10);
      for (double n = 1; n < 20; n += 2)
            product *= n;
      cout << "\nThe reciprocal of 1*3*5* ... * 19 = " << 1/product << endl;
}
```
Output:

The reciprocal of 1*3*5* ... * 19 = 1.527349309e-009

c. Fill up an array of 3 elements with 3 random numbers between 1 and 10 and then compute the cube root of their product. Do not use the `cmath` library.

```
int main()
{
      srand(time(NULL));
      double randDoubles[3], product = 1;
      for(int i = 0; i < 3; ++i)
            randDoubles[i] = (double)(1+rand()%10);
      cout << setprecision(10);
      for (int n = 0; n < 3; n++)
            product *= randDoubles[n];
      cout << "\nThe cube root of the random product " << product << " is "
             << croot(product) << endl;
}

double croot(double A)
{     // Use the babylonian algorithm to compute the square root
      double x1 = A, x2 = x1/2;
      const double tolerance = 1e-14;
      while(x2 - x1 > tolerance || x1 - x2 > tolerance)
      {
            x1 = x2;
            x2 = (2*x1 + A/(x1*x1))/3;
      }
      return x2;
}
```
Output:
The cube root of the random product 40 is 3.419951893

d. Compute the value of the zeta function, $\zeta(s) = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \cdots$ for any positive integer $s > 1$.

```
int main()
{
      double input;
```

```
        cout << setprecision(10);
        cout << "\nzeta(2) = " << zeta(2) << " = " << PI*PI/6. << endl;
        cout << "\nEnter another input: ";
        cin >> input;
        cout << "\nzeta(" << input << ") = " << zeta(input) << endl;
}

double zeta(double s) {
        double zetaVal = 1, term = 1., n = 1.;
        while(term > 1.e-15)
        {
                ++n;
                term = 1.;
                for(double pow = 1;pow <= s; ++pow)
                        term *= 1/n;
                zetaVal += term;
        }
        return zetaVal;
}

Output:

zeta(2) = 1.644934035 = 1.644934067

Enter another input: 5

zeta(5) = 1.036927755
```

11. Consider the following program fragment.

```
int accumulator = 0, sam, pam;
cout << "\nEnter integers for sam and pam: ";
cin >> sam >> pam;
while (true ) {
    if (pam == 0) break ;
    accumulator += ((pam % 2 == 1) ? sam : 0);
    pam /= 2;
       sam *= 2;
}
cout << accumulator << "\n";
```

a. Complete the following tables until the program completes, or indicate that it's an infinite loop:

sam	pam	accumulator
5	4	0
10	2	0
20	1	0
40	0	20

sam	pam	accumulator
6	17	0
12	8	6
24	4	6
48	2	6
96	1	6
192	0	102

b. In a few words, describe the output of this program in terms of the input values of `sam` and `pam`, and how it works.

ANS: This is a multiplication algorithm that uses the binary form of the second factor like so: $5 * 4 = 5 * (2^2) = 20$ so just double 5 twice. This is recognized by dividing by writing $2^2 = 100_2$ and proceding thus: $5 * 100_2 = 10 * 10_2 = 20 * 1 = 20$. Similarly,

$$6 * 17 = 6 * 10001_2 = 6 + 12 * 1000_2 = 6 + 24 * 100_2 = 6 + 48 * 10_2 = 6 + 96 * 1 = 102$$

or $3 * 21 = 3 * 10101_2 = 3 + 6 * 1010_2 = 3 + 12 * 101_2 = 15 + 24 * 10_2 = 15 + 48 = 63.$

12. Suppose that p = 0x0012FF50, &a = 0x0012FF44, &p = 0x0012FF38.
What will be output by the following code?

```
int main()
{ int a[] = { 22, 33, 44, 55, 66, 77, 88, 99 };
  int* p = &a[3];                              // p points to a[3]
  cout << "p = " << p << ", *p = " << *p;
  cout << "\n&a = " << &a;                    // ok: a is an lvalue
  cout << "\n&p = " << &p;                    // ok: p is an lvalue
  cout << "\n&(a[5]) = " << &(a[5]);       // ok: a[5] is an lvalue
  cout << "\n&(*(a+5)) = " << &(*(a+5));// ok: *(a+5) is an lvalue
  cout << "\n&(*(p+2)) = " << &(*(p+2));// ok: *(p+2) is an lvalue
  cout << "\n&(p+2) = " << p+2; //&(p+2);  // ERROR: p+2 is not an lvalue
  cout << endl;
}
```

SOLN: (note this is not really fair game for the exam)

```
p = 0012FF50, *p = 55
&a = 0023FF44
&p = 0012FF38
&(a[5]) = 0023FF58
&(*(a+5)) = 0023FF58
&(*(p+2)) = 0023FF58
&(p+2) = 0023FF58
```

13. Write a C++ program to implement the flow chart below.

```
// implementation of flow chart

#include <iostream>

using namespace std;

int main()
{
     int b,n,z = 1;
     cout << "\nInput b and n and we'll compute
          << something with them: ";
     cin >> b >> n;
     cout << "\nb = " << b << "\nn = " << n;
     while(n!=0) {
           if(n%2==0) // n is even
           {
                 --n;
                 z *= b;
           }
           else
           {
                 b *= b;
                 n /= 2;
           }
     }
     cout << "\nThe result is " << b << endl;
}
```