Thursday, June 27, 2013.

Programming Fundamentals – Spring 2013
(BS-SE-F12 Morning & Afternoon)

Lab # 15

Instructions:

- Indent your code properly.
- Use meaningful variable and function names. Follow the naming conventions.
- Use meaningful prompt lines/labels for all input/output that is done by your programs.

Task # 0

Type the following program (do NOT copy-paste), and execute it to understand its working. You will have to execute it from the command prompt to specify the arguments.

```
// This program demonstrates how to read command line arguments.
#include <iostream>
using namespace std;

int main (int argc, char *argv[])
{
    cout << "You entered " << (argc - 1);
    cout << " command line arguments.\n";
    if (argc > 1)
    {
        cout << "Here they are:\n";
        for (int count = 1; count < argc; count++)
        {
            cout << argv[count] << endl;
        }
    }
    return 0;
}
```

Task # 1

Now write a C++ program which takes one or more integers from the command line and displays the largest of these integers on screen. Few sample executions of this program will look like as follows (here `largest` is the name of the executable file of your program). *Note: Text shown in red is entered by the user at command prompt.*

```
C:\> largest 3 5
You entered 2 number(s).
The largest number is 5.

C:\> largest
ERROR: You must enter at least one number.

C:\> largest 2 5 7 -4 3
You entered 5 number(s).
The largest number is 7.
```

Task # 2

Type the following function (do NOT copy-paste) and execute it on some inputs to understand its working:

```
// Function to print the bit pattern of a given signed integer
void bitPattern (int u)
{
    int i, x, word;
    unsigned int mask;

    mask = 1;
    word = 8 * sizeof (int);
    mask = mask << (word-1);

    for (i=1; i<=word; i++)
    {
        if (u & mask)
            x = 1;
        else
            x = 0;

        cout << x;

        mask = mask >> 1;

        if (i%8==0)
            cout << " ";
    }
    cout << endl;
}
```

Task # 3

Implement a function **countOnes** that counts the number of 1's in a signed integer. A sample main function illustrating the usage of this function is given below:

```
int main()
{
    int count = countOnes(5);
    cout << count << endl;  // should display 2
    return 0;
}
```

Task # 4

Implement a function **invertBits** which receives an unsigned character and changes all 0's to 1's and all 1's to 0's. A sample main function illustrating the usage of this function is given below:

```
int main()
{
    unsigned char c = invertBits(67); // bit pattern of 67 is 01000011
    cout << static_cast <int> c; // should display 188 (bit pattern: 10111100)
    return 0;
}
```

Task # 5

Implement a function:

```
    void setBit (int& n, int j)
```

to set (1) the **j**th bit of **n** starting from the right-most (least-significant) bit. The right-most bit considered to be bit # 1, the next bit is considered to be bit # 2, and so on. Test the correctness of your implemented function by using the **bitPattern** function implemented in Task # 2.

Task # 6

Implement a function:

```
    void unsetBit (int& n, int j)
```

to unset (0) the **j**th bit of **n** starting from the right-most (least-significant) bit. The right-most bit considered to be bit # 1, the next bit is considered to be bit # 2, and so on. Test the correctness of your implemented function by using the **bitPattern** function implemented in Task # 2.