CSC 600 Homework **Dr. Jozo Dujmović**
Procedural programming

Programming is similar to a game of golf.
The point is not getting the ball in the hole
but how many strokes it takes.
- H. Mills

Inside every well-written large program
is a well-written small program.
- C.A.R. Hoare

The goal of this homework is to develop your skills in procedural programming. You should develop four efficient procedural programs and perform a simple performance measurement. Your programs should be as short and simple as possible, and as fast as possible. Each program must be documented. For each solution first explain the idea and then explain the reasons why you think it is a good procedural solution. You must show your source programs, and results of their execution. All programs should be written in C++.

1. Plateau program (max sequence length) (a combinatorial algorithm)

The array **a(1..n)** contains sorted integers. Write a function **maxlen(a,n)** that returns the length of the longest sequence of identical numbers (for example, if a=(1,1,1,2,3,3,5,6,6,6,6,7,9) then **maxlen** returns 4 because the longest sequence 6,6,6,6 contains 4 numbers. Write a demo main program for testing the work of maxlen. Explain your solution, and insert comments in your program. The solution should have time complexity O(n).

2. Integer plot function (find a smart way to code big integers)

Write a program **BigInt(n)** that displays an arbitrary positive integer n using big characters of size 7x7, as in the following example for **BigInt(170):**

```
  @@    @@@@@@@   @@@@@
 @@@         @@  @@   @@
  @@        @@   @@   @@
  @@       @@    @@   @@
  @@      @@     @@   @@
  @@     @@      @@   @@
@@@@@@  @@        @@@@@
```

Write a demo main program that illustrates the work of BigInt(n) and prints the following sequence of big numbers 1, 12, 123, 1234, …, 1234567890, one below the other.

3. Array processing (elimination of three largest values) (one of many array reduction problems)

The array **a(1..n)** contains arbitrary integers. Write a function **reduce(a,n)** that reduces the array **a(1..n)** by eliminating from it all values that are equal to three largest different integers. For example, if a=(9,1,1,6,7,1,2,3,3,5,6,6,6,6,7,9) then three largest different integers are 6,7,9 and after reduction the reduced array will be a=(1,1,1,2,3,3,5), n=7. The solution should have time complexity O(n).

4. Iteration versus recursion (an opportunity for performance measurement)

Make a sorted integer array `a[i]=i`, i=0,…,n-1. Let **`bs(a,n,x)`** be a binary search program that returns the index i of array a[0..n-1] where a[i]=x. Obviously, the result is **`bs(a,n,x)=x`**, and the binary search function can be tested using the loop

```
for(j=0; j<K; j++)
     for(i=0; i<n; i++) if(bs(a,n,i) != i) cout << "\nERROR";
```

Select the largest **n** your software can support and then **K** so that this loop with an iterative version of **bs** runs 3 seconds or more. Then measure and compare this run time and the run time of the loop that uses a recursive version of **bs**. Compare these run times using maximum compiler optimization (release version) and the slowest version (minimum optimization or the debug version). If you use a laptop, make measurements using AC power, and then same measurements using only the battery. What conclusions can you derive from these experiments? Who is faster? Why?

5. Iteration versus recursion (another opportunity for performance measurement)

Write a recursive function Frec(n) that computes Fibonacci numbers. Then write an iterative version of Fibonacci number function Fit(n). Functions Frec(n) and Fit(n) return the same value but with different performance.

Write the main program that discovers the value N10 so that Frec(N10) runs on your machine 10 seconds (or approximately 10 seconds). Then measure the run time of Fit(N10) and compute how many times is Fit(N10) faster than Frec(N10). Show what is N10 on your machine.

Notes:

1. When you measure the speed, your machine should be disconnected from the Internet, it should use the AC power supply, and it should run only one program (your performance measurement program).
2. In C++ you can measure current time in seconds using the following function:

   ```
   double sec(void)
   {
        return double(clock())/double(CLOCKS_PER_SEC);
   }
   ```

   To measure the run time of fast programs you must repeat them many times inside a loop. Take care to eliminate the overhead generated by the loop.