CSc 600-01 (Section 1)
Homework 2 - Procedural Programming
prepared by Ilya Kopyl

CSc 600 Homework 2 - Procedural Programming

February 27, 2018

Homework is prepared by: Ilya Kopyl.
It is formatted in LaTeX, using TeXShop editor (under GNU GPL license).

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 time complexity of the solution should be in O(n).

The answer is listed on the pages 2 through TBD.

A code listing of implementation of maxlen function:

```c
unsigned int maxlen(int *a, unsigned int n)
{
    // handling the edge cases - arrays of size 0 and 1:
    if (n < 2)
        return n;

    unsigned int max_count, current_count, i;
    i = max_count = 0;
    current_count = 1;

    printf("    a[%d]=%d; \tcurrent_count=%d; \tmax_count=%d\n",
            i, a[i], current_count, max_count);

    for (i = 1; i < n; ++i)
    {
        if (a[i] == a[i-1])             // counting the current sequence
        {
            current_count++;

            // checking whether the longest sequence is at the end of array
            if(i == n-1 && current_count > max_count)
                max_count = current_count;
        }
        else                            // starting the count of the new sequence
        {
            // before resetting the counter, save it's value if it is above threshold
            if (current_count > max_count)
                max_count = current_count;

            // exit the loop if max_count is sufficiently large
            if (max_count >= n-i)
                break;

            current_count = 1;
        }

        printf("    a[%d]=%d; \tcurrent_count=%d; \tmax_count=%d\n",
                i, a[i], current_count, max_count);
    }
    return max_count;
}
```

The result of the program execution:

```
Array a:    1  1  1  2  3  3  5  6  6  6  6  7  9
    a[0]=1;      current_count=1;       max_count=0
    a[1]=1;      current_count=2;       max_count=0
    a[2]=1;      current_count=3;       max_count=0
    a[3]=2;      current_count=1;       max_count=3
    a[4]=3;      current_count=1;       max_count=3
    a[5]=3;      current_count=2;       max_count=3
    a[6]=5;      current_count=1;       max_count=3
    a[7]=6;      current_count=1;       max_count=3
    a[8]=6;      current_count=2;       max_count=3
    a[9]=6;      current_count=3;       max_count=3
    a[10]=6;     current_count=4;       max_count=3
Max sequence length of array a = 4

Array b:
Max sequence length of array b = 0

Array c:    12
Max sequence length of array c = 1

Array d:    16  16  16  18  18  20
    a[0]=16;     current_count=1;       max_count=0
    a[1]=16;     current_count=2;       max_count=0
    a[2]=16;     current_count=3;       max_count=0
Max sequence length of array d = 3

Array e:    0   0
    a[0]=0;      current_count=1;       max_count=0
    a[1]=0;      current_count=2;       max_count=2
Max sequence length of array e = 2

Array f: 0   1
    a[0]=0;      current_count=1;       max_count=0
Max sequence length of array f = 1

Array g:    1  2  3  3
    a[0]=1;      current_count=1;       max_count=0
    a[1]=2;      current_count=1;       max_count=1
    a[2]=3;      current_count=1;       max_count=1
    a[3]=3;      current_count=2;       max_count=2
Max sequence length of array g = 2
```

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):

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

The answer is listed on the pages TBD through TBD.

4

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 would be a=(1, 1, 1, 2, 3, 3, 5), n=7. The time complexity of the solution should be in O(n).

The answer is listed on the pages TBD through TBD.

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 the 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

```cpp
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 the same measurements using only the battery. What conclusions can you derive from these experiments? Who is faster? Why?

The answer is listed on the pages TBD through TBD.

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

Following is