2S03 Assignment 1

Ned Nedialkov

21 September 2018

Due: 3 October in class

Problem 1 (5 points) You are asked to print n, 3×3 squares to the screen. A 3×3 square must be represented by the following ASCII image:

```
###
###
###
```

There must be at most 5 squares per row, one blank space between adjacent squares, and one blank line between rows of squares. Also, a row must be filled entirely before the next one starts. For example, your input and output should look like

```
Enter number of squares: 7
### ### ### ### ###
### ### ### ### ###
### ### ### ### ###

### ###
### ###
### ###
```

If the user enters $n \leq 0$ or $n > 50$, your program should ask again for input.

Store your program in a file with name `squares.c`.

Problem 2 (5 points) The inverse square root of $a > 0$ is defined as $1/\sqrt{a}$.

One way to compute the inverse square root is to use Newton's method. Given $a > 0$ and an initial guess $x_0 > 0$ for $1/\sqrt{a}$, we compute using

$$x_{i+1} = x_i(1.5 - 0.5ax_i^2), \quad i = 0, \ldots, n-1. \tag{1}$$

Note that this includes only $*$ and $-$.

Write a program that takes as input a, x_0, and n and outputs on each line the iteration number $i+1$, the computed approximation x_i, and its relative error $e_i = |x_i - 1/\sqrt{a}|/(1/\sqrt{a}) = |x_i - 1/\sqrt{a}|\sqrt{a}$.

For example my input/output looks like

```
Enter a, x0, and n: 2 0.1 10
  1 1.490000000000000e-01  7.89e-01
  2 2.201920510000000e-01  6.89e-01
  3 3.196121663445493e-01  5.48e-01
  4 4.467692476757679e-01  3.68e-01
  5 5.809774962956584e-01  1.78e-01
  6 6.753660916796385e-01  4.49e-02
  7 7.050015895089734e-01  2.98e-03
  8 7.070973891810239e-01  1.33e-05
  9 7.071067809994271e-01  2.65e-10
 10 7.071067811865476e-01  1.57e-16
```

Store your program in file `invsq.c`.

Problem 3 (3 points) In the previous problem, use as an initial approximation $x_0 = 2/(1 + a)$. Write a program that reads only a and stops when the error is $\leq 10^{-13}$. It should output the computed approximations and the corresponding errors, e.g.

```
Enter a: 3
 5.625000000000000e-01    2.57e-02
 5.767822265625000e-01    9.84e-04
 5.773494311370087e-01    1.45e-06
 5.773502691878011e-01    3.16e-12
 5.773502691896257e-01    1.92e-16
```

Store your program in file `invsq2.c`.

Problem 4 (5 points) Write a program that asks the user for a positive integer, say n, creates a Fibonacci sequence up to n, and counts all numbers in this sequence that are divisible by 2, 3, 5, 6, 10, and 15, and the numbers that are not divisible by any of the above numbers.

Output the result in a table, e.g.

```
Enter a positive integer : 2000

Fibonacci sequence up to 2000 :
 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597,

 divisibley by:
 2                5
 3                4
 5                3
 6                1
10                1
15                0
 -                7
```

Store your program in file `countfib.c`.

Submit

- Hard copy of your C files.
- The output for each of your programs.
- The C files to svn at `https://websvn.cas.mcmaster.ca/se2s03/macid/A1`. `macid` is the one you use in `macid@mcmaster.ca`

  A1 means create a subdirectory with name A1 and submit your files into this subdirectory.
- If you have not used SVN, read Subversion Tutorial: 10 Most Used SVN Commands with Examples

Comments

- Follow the instructions precisely. You will lose marks even if your file naming is not as specified.
- Test your programs. Marks will be deducted if for some inputs your program(s) does not produce correct output.
- If a program does not compile, the corresponding mark will be zero.
- Submit $\boxed{\text{your own work}}$. All submitted programs will be checked for similarities.