**Task 1: Create a brief introduction to an historic Computer Scientist**

Introduction of Grace Hooper

Rui Ma

The Practice of Computer Science

Professor Mary Sanseverino

September 20, 2014

Introduction of Grace Hooper

I’m glad to be given the opportunity to introduce a great computer scientist Grace Hopper.

Grace Murray Hopper was the first female American Navy rear admiral and one of the most famous computer programming scientists. She expanded the concept of machine-independent programming by developing COBOL, a streamlined and standardized business-oriented programming language. This led her to be known as “the mother of COBOL,” (Abbate, p.215) the first modern programming language. She also contributed to the use of specialized computer science terminology. For example, she is well known for popularizing the use of the word “debugging” when solving computer errors (Rajaraman, 2001).

Most importantly, she was the first scientist who completed the first compiler of A-0 System (Arithmetic Language version 0), which can compile programs to machine codes. The A-0 system later became a precursor for the modern day linker or loader (George, 2014). Also, she was the first to notice the Millennium bug (Y2K) which was an issue of computer related, non-digital documentation and data storage. In addition to these achievements, she successfully mentored many famous programming language experts throughout her lifetime.

References

Abbate, J. (2011). *Software’s founding mother: Kurt W. beyer: Grace hopper and the invention of the information age.* Cambridge, mass.: MIT press, 2009, 408 pp. Dordrecht: Springer Netherlands. doi:10.1007/s11016-010-9418-z

Georgi, D.(2014). *First Compiler of Grace Hopper.* Retrieved from: http://history-computer.com/ModernComputer/Software/FirstCompiler.html

Rajaraman, V. (2001). *Grace murray hopper — programming pioneer.* Resonance, 6(2), 2-3. doi:10.1007/BF02836935

Task 2: **Analyzing an algorithm.**

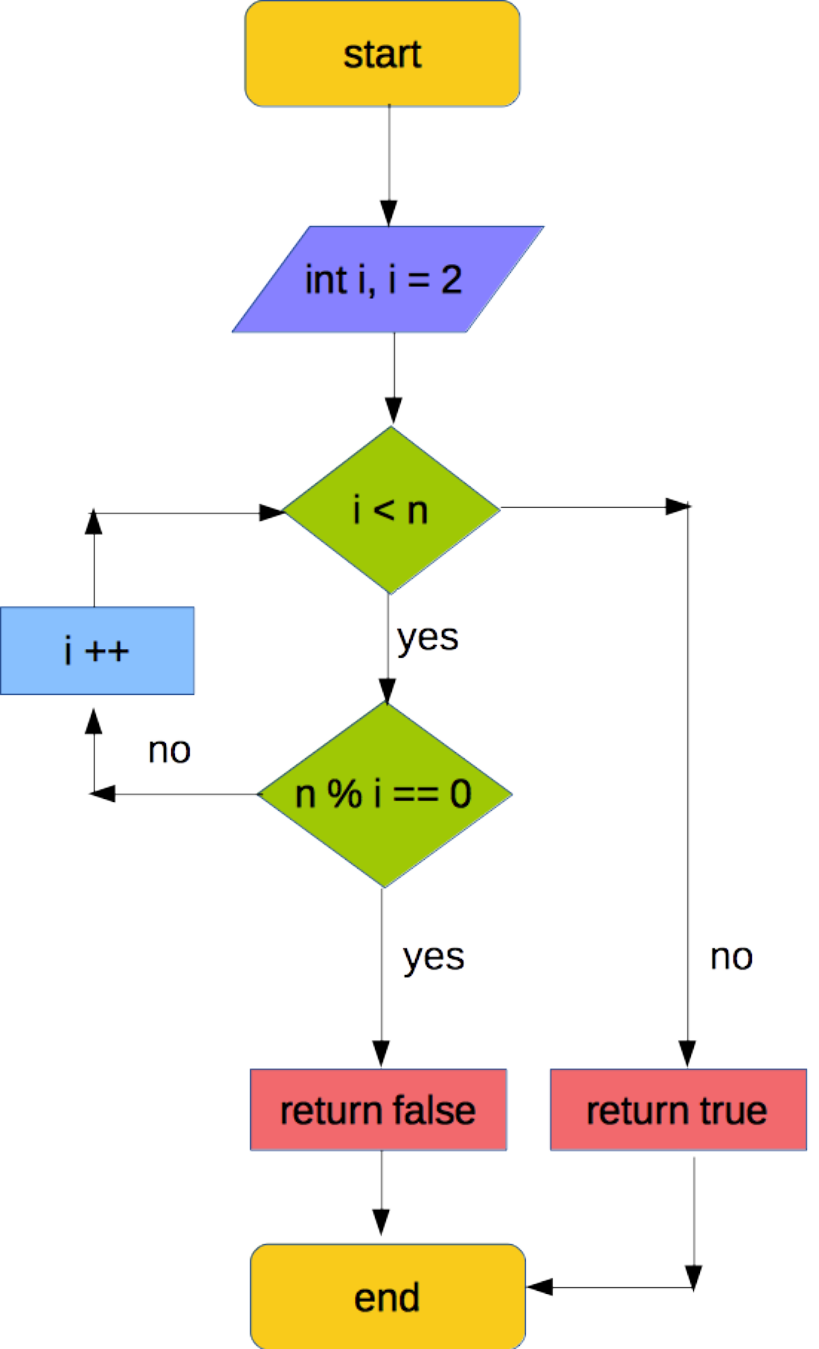
The recipe, as an algorithm, can be analyzed from five aspects. Firstly, from perspective of finiteness, there is no stop command after every steps and the whole procedure. Secondly, from the perspective of definiteness, it does not define the extent of “slight beaten egg” in step 1. In step 2, it doesn’t define “mixture”, and it should give a definition about how to mix. In step 3, it does not define the amount of “butter” and “golden brown”. In order to fry something, it should define the frying condition: temperature of oil, and frying time. Thirdly, from the perspective of input, at the beginning of recipe, all the materials should be listed. Fourthly, from the perspective of output, there is no conclusion at the end of the four steps. Fifthly, from the perspective of effectiveness, the recipe should define operate time.

**Task 3: Determine if a given positive integer is a prime number.**

Precondition: Assume that all data coming to the algorithm has already been checked and is indeed valid integer data. Assume the integer is, at most, 5000.

Set n is the given positive integer and n < 5000

start

**ineteminet. println﷽﷽﷽﷽﷽﷽﷽﷽﷽ineteminet. println﷽﷽﷽﷽﷽﷽﷽﷽﷽**

get n

for (int i = 2; i < 5000; i ++);

if ( n % i ) == 0

then

return false

end if

end for

return true

end

**Task 4: Some binary and hex fun.**

**1) How to arrange the candles and make them represent 57 in binary (base 2)**

**57(base 10) = 111001(base 2)**

****

Happy Birthday Mary

**2)** What is Mary’s age (57) in hexadecimal (base 16)?

Mary is 39-year-old in hexadecimal.

3) Fill in the blank.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hex | Binary value in 8 bits | Integer - unsigned | Integer – signed magnitude | Integer - 2’s compliment |
| B116 | 10110001 | 177 | -49 | -79 |
| 1B16 | 00011011 | 27 | 27 | 27 |

**4)** Convert the following to Binary (2’s complement) using 8 bits

|  |  |
| --- | --- |
| Signed decimal | Convert to Binary (2’s complement) using 8 bits |
| -22 | 11101010 |
| 8 | 00001000 |
| -122 | 10000110 |
| 39 | 00100111 |