COSC 600
Programming Assignment 1

Due date : Oct. 3(Tuesday) by 11:59 pm

<u>Problem 1 (Guessing Game)</u>

Write a program to find the integer number which was generated by random number function ranged from 1 to 1,000,000. Your program needs to use binary search concept and your program will answer “Lower”, “Higher”, or “Found!”. Your program has to repeat this guess game for 100 times and calculate the number of guessing trial for each random number. And calculate the average number of guessing trial for 100 random numbers.

Example)

```
Guess-1:
The number is lower than 500000
The number is higher than 750000
.
.
.
The number is found. It is xxxxx.
The number of guessing trial is yy.

Guess-2:
(the output format is same as Guess-1 case.)

.
.
.
```

<u>Problem 2</u>

Write a program to convert a decimal number into a binary number using a recursive function. (Do not use an array or a stack.)

Example)

```
Enter a positive decimal number  :  57

The binary number is  1 1 1 0 0 1
```

Problem 3 Word puzzle problem
Word puzzle problem is to find the words in a two-dimensional array of characters. The input to this problem is two input files: one is a two-dimensional array of characters and the other is a list of words we will find in two-dimensional array. These words may be horizontal, vertical, or diagonal in any direction (for a total of eight directions).
Let's look at a simple example with chars:
A 2-D 7x7 puzzle and a word list file are as follows:

```
Puzzle                        word file                     Output
n o h t y p s         ruby
m i a r y c c         cave                                   c
l l e k s a h                                               a
r u b y v m e                                   r u b y v
e h h e l l m                                 e
p c j n i c e
r e e k b i p
```

Write a program to find for this problem and run your program with two given input data files (50 x 50 puzzleinput.txt and a list of words, wordlist.txt.). And measure the actual running times of your program.

Problem 4

A majority element in an array, A, of size N is an element that appears more than N/2 times(thus, there is at most one). If there is no majority element, your program should indicate this. Design two different algorithms and write them in Java, C++, or Python.
And then run these algorithms with four given sample sets(Majex1, 2, 3, 4) of input files and measure execution time for each case.

a) Method 1 (O(NlogN) algorithm using a divided-and-conquer method. The algorithm begins by splitting the array in half repeatedly and calling itself on each half. When we get down to single elements, that single element is returned as the majority of its (1-element) array. At every other level, it will get return values from its two recursive calls.
The pseudo-code of the algorithm is as follows:

```
procedure GetMajorityElement(a[1...n])
Input: Array a of objects
Output: Majority element of a
if n = 1: return a[1]
k = n/2
elemlsub = GetMajorityElement(a[1...k])
elemrsub = GetMajorityElement(a[k+1...n]
if elemlsub = elemrsub:
return elemlsub
```

```
lcount = GetFrequency(a[1...n],elemlsub)
rcount = GetFrequency(a[1...n],elemrsub)
if lcount > k+1:
return elemlsub
else if rcount > k+1:
return elemrsub
else return NO-MAJORITY-ELEMENT
```

b) Method 2 (O(N)) algorithm

This algorithm is a two step process.

1. Get an element occurring most of the time in the array. This phase will make sure that if there is a majority element then it will return that only.
2. Check if the element obtained from above step is majority element.

The following is pseudo code of step 1.

```
// Initialize index and count of majority element
Maj_index = 0, count = 0
Loop for i = 0 to N – 1
{
    if  a[Maj_index] = = a[i]
        count++
    else
        count - -
    if count = = 0
        Maj_index = i
        count = 1
}
return a[Maj_index]
```