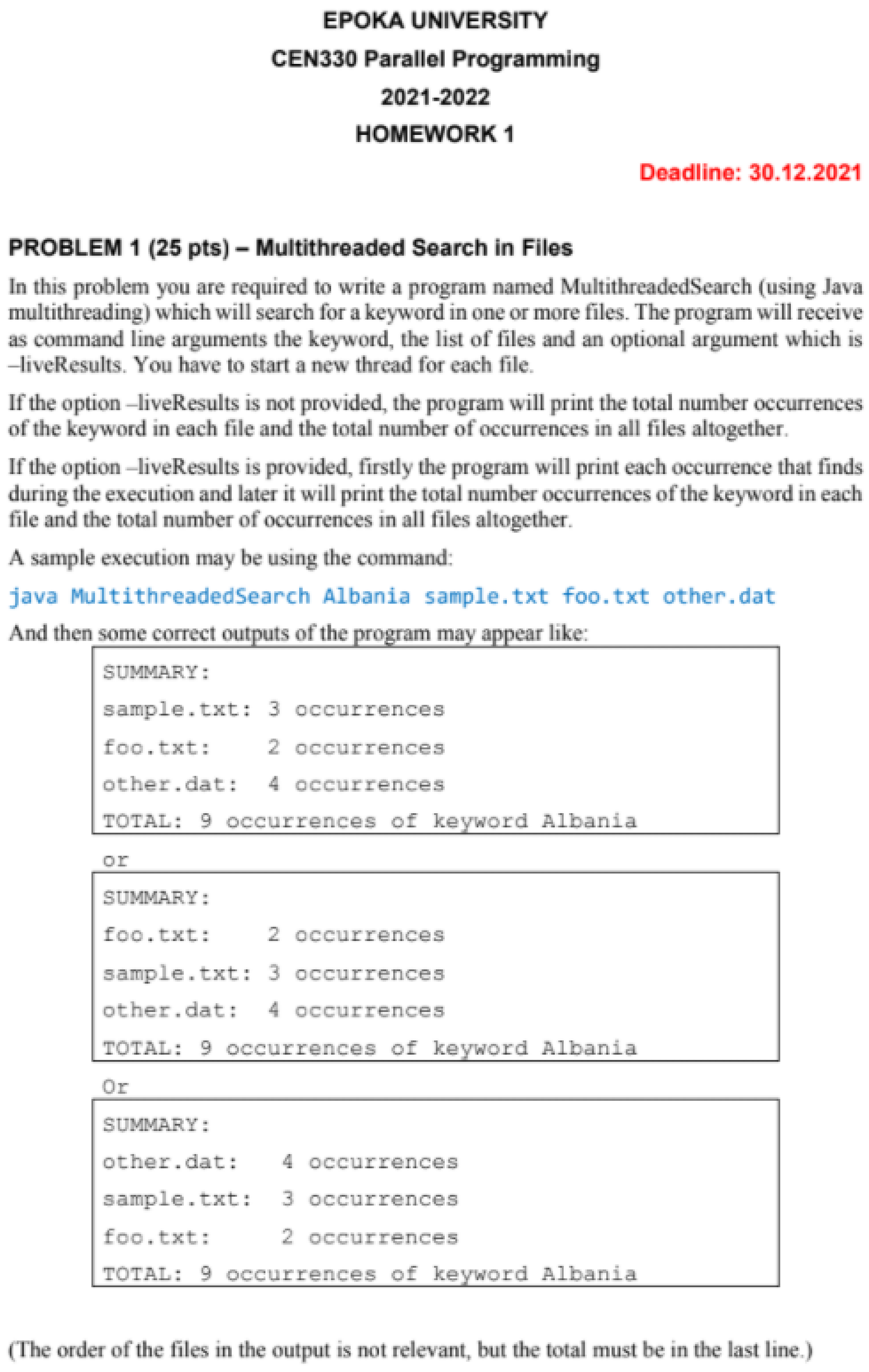


EPOKA UNIVERSITY

CEN330 Parallel Programming

2021-2022

HOMEWORK 1

Deadline: 30.12.2021

PROBLEM 1 (25 pts) – Multithreaded Search in Files

In this problem you are required to write a program named MultithreadedSearch (using Java multithreading) which will search for a keyword in one or more files. The program will receive as command line arguments the keyword, the list of files and an optional argument which is –liveResults. You have to start a new thread for each file.

If the option –liveResults is not provided, the program will print the total number occurrences of the keyword in each file and the total number of occurrences in all files altogether.

If the option –liveResults is provided, firstly the program will print each occurrence that finds during the execution and later it will print the total number occurrences of the keyword in each file and the total number of occurrences in all files altogether.

A sample execution may be using the command:

```
java MultithreadedSearch Albania sample.txt foo.txt other.dat
```

And then some correct outputs of the program may appear like:

```
SUMMARY:
sample.txt: 3 occurrences
foo.txt:    2 occurrences
other.dat:  4 occurrences
TOTAL: 9 occurrences of keyword Albania
```

or

```
SUMMARY:
foo.txt:    2 occurrences
sample.txt: 3 occurrences
other.dat:  4 occurrences
TOTAL: 9 occurrences of keyword Albania
```

Or

```
SUMMARY:
other.dat:   4 occurrences
sample.txt:  3 occurrences
foo.txt:     2 occurrences
TOTAL: 9 occurrences of keyword Albania
```

(The order of the files in the output is not relevant, but the total must be in the last line.)

Edison Ponari
CEN 330 / CEN A

Another sample execution may be using the command:

```
java MultithreadedSearch Albania sample.txt foo.txt other.dat -liveResults
```

And then some correct outputs of the program may appear like:

```
LIVE RESULTS:
sample.txt:  1 occurrence  at line 15
other.dat:   2 occurrences at line 13
sample.txt:  1 occurrence  at line 24
foo.txt:     1 occurrence  at line 18
other.dat:   1 occurrence  at line 39
sample.txt:  1 occurrence  at line 95
foo.txt:     1 occurrence  at line 68
other.dat:   1 occurrence  at line 62
----------------------------------------------
SUMMARY:
sample.txt:  3 occurrences
foo.txt:     2 occurrences
other.dat:   4 occurrences
TOTAL: 9 occurrences of keyword Albania
```

or

```
LIVE RESULTS:
foo.txt:     1 occurrence  at line 18
sample.txt:  1 occurrence  at line 15
sample.txt:  1 occurrence  at line 24
other.dat:   2 occurrences at line 13
foo.txt:     1 occurrence  at line 68
sample.txt:  1 occurrence  at line 95
other.dat:   1 occurrence  at line 39
other.dat:   1 occurrence  at line 62
----------------------------------------------
SUMMARY:
sample.txt:  3 occurrences
foo.txt:     2 occurrences
other.dat:   4 occurrences
TOTAL: 9 occurrences of keyword Albania
```

(The order of the lines in the live results is not relevant, but the summary must appear after the live results. Also, the order of the files in the summary is not relevant, but the total must be in the last line.)

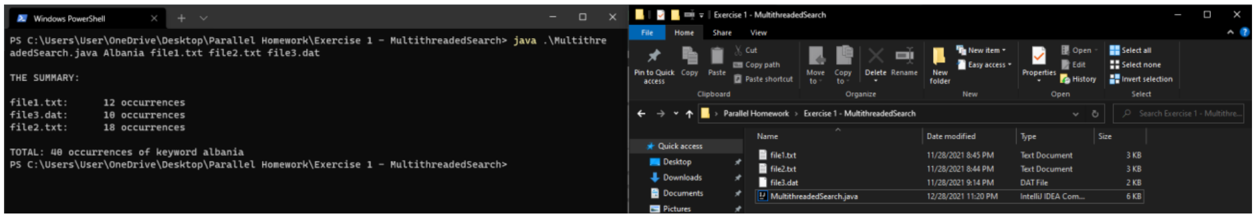
Solution:

Source code is provided in "Exercise 1 - MultithreadedSearch" folder. I wrote a program that will take each file path and create a thread for each of them. They are going to count how many times the word is repeated in the file. We need to print the summary before printing the results so I came up with this solution. After starting every thread with their path of the file to read, I join all the threads together and after they are finish, I print the summary line and create different threads to print the results of each thread. In the liveResults mode, I just make sure that each time an occurrence happens, it is printed immediately.

The source code is well documented with comments.

Java Multithreading:

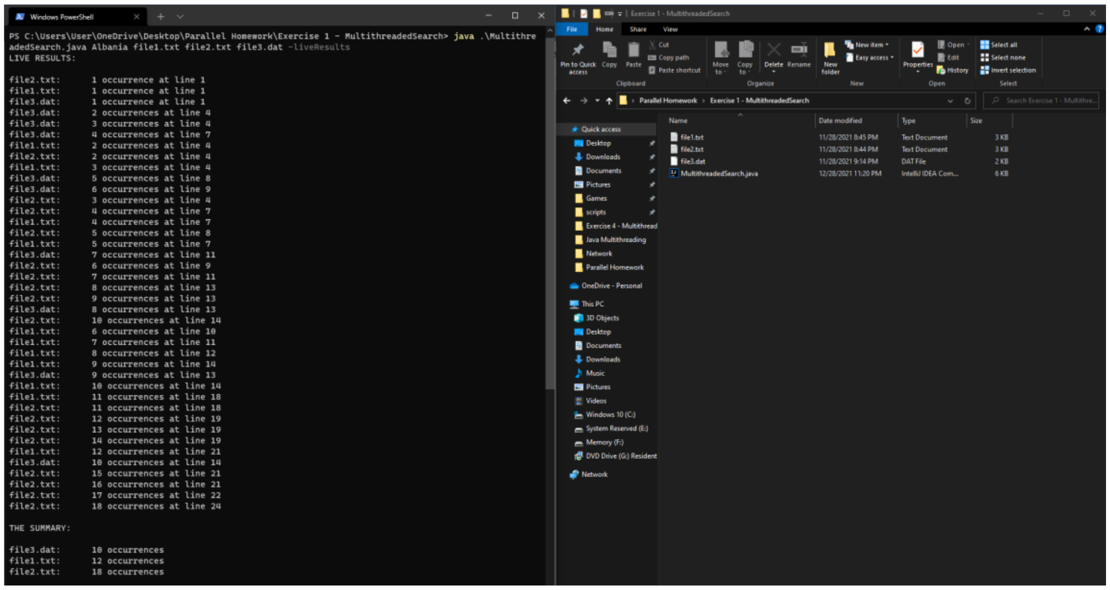
•Without -liveResults:


Windows PowerShell
Exercise 1 - MultithreadedSearch
File
Home
Share
View
Parallel Homework
Exercise 1 - MultithreadedSearch

```
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 1 - MultithreadedSearch> java .\Multithre
adedSearch.java Albania file1.txt file2.txt file3.dat

THE SUMMARY:

file1.txt:      12 occurrences
file3.dat:      10 occurrences
file2.txt:      18 occurrences

TOTAL: 40 occurrences of keyword albania
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 1 - MultithreadedSearch>
```

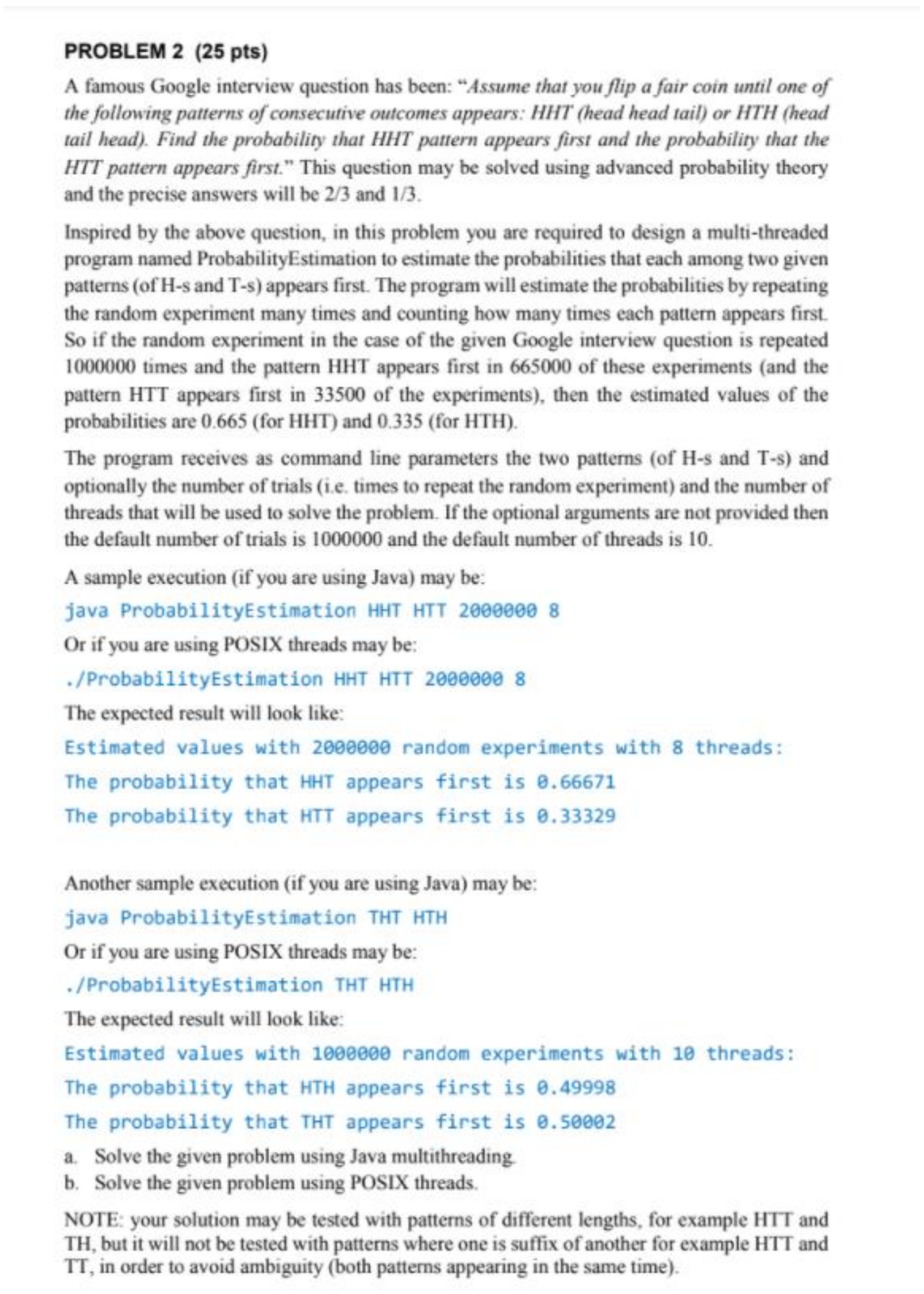
Name	Date modified	Type	Size
file1.txt	11/28/2021 8:45 PM	Text Document	3 KB
file2.txt	11/28/2021 8:44 PM	Text Document	3 KB
file3.dat	11/28/2021 9:14 PM	DAT File	2 KB
MultithreadedSearch.java	12/28/2021 11:20 PM	IntelliJ IDEA Com...	6 KB

•With -liveResults:


Windows PowerShell
Exercise 1 - MultithreadedSearch
File
Home
Share
View
Parallel Homework
Exercise 1 - MultithreadedSearch

```
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 1 - MultithreadedSearch> java .\Multithre
adedSearch.java Albania file1.txt file2.txt file3.dat -liveResults
LIVE RESULTS:

file2.txt:      1 occurrence at line 1
file1.txt:      1 occurrence at line 1
file3.dat:      1 occurrence at line 1
file3.dat:      2 occurrences at line 4
file3.dat:      3 occurrences at line 4
file3.dat:      4 occurrences at line 7
file1.txt:      2 occurrences at line 4
file2.txt:      2 occurrences at line 4
file1.txt:      3 occurrences at line 4
file3.dat:      5 occurrences at line 8
file3.dat:      6 occurrences at line 9
file2.txt:      3 occurrences at line 4
file2.txt:      4 occurrences at line 7
file1.txt:      4 occurrences at line 7
file2.txt:      5 occurrences at line 8
file1.txt:      5 occurrences at line 7
file3.dat:      7 occurrences at line 11
file2.txt:      6 occurrences at line 9
file2.txt:      7 occurrences at line 11
file2.txt:      8 occurrences at line 13
file2.txt:      9 occurrences at line 13
file3.dat:      8 occurrences at line 13
file2.txt:      10 occurrences at line 14
file1.txt:      6 occurrences at line 10
file1.txt:      7 occurrences at line 11
file1.txt:      8 occurrences at line 12
file1.txt:      9 occurrences at line 14
file3.dat:      9 occurrences at line 13
file1.txt:      10 occurrences at line 14
file1.txt:      11 occurrences at line 18
file2.txt:      11 occurrences at line 18
file2.txt:      12 occurrences at line 19
file2.txt:      13 occurrences at line 19
file2.txt:      14 occurrences at line 19
file1.txt:      12 occurrences at line 21
file3.dat:      10 occurrences at line 14
file2.txt:      15 occurrences at line 21
file2.txt:      16 occurrences at line 21
file2.txt:      17 occurrences at line 22
file2.txt:      18 occurrences at line 24

THE SUMMARY:

file3.dat:      10 occurrences
file1.txt:      12 occurrences
file2.txt:      18 occurrences
```

Name	Date modified	Type	Size
file1.txt	11/28/2021 8:45 PM	Text Document	3 KB
file2.txt	11/28/2021 8:44 PM	Text Document	3 KB
file3.dat	11/28/2021 9:14 PM	DAT File	2 KB
MultithreadedSearch.java	12/28/2021 11:20 PM	IntelliJ IDEA Com...	6 KB



PROBLEM 2 (25 pts)

A famous Google interview question has been: "*Assume that you flip a fair coin until one of the following patterns of consecutive outcomes appears: HHT (head head tail) or HTH (head tail head). Find the probability that HHT pattern appears first and the probability that the HTT pattern appears first.*" This question may be solved using advanced probability theory and the precise answers will be 2/3 and 1/3.

Inspired by the above question, in this problem you are required to design a multi-threaded program named ProbabilityEstimation to estimate the probabilities that each among two given patterns (of H-s and T-s) appears first. The program will estimate the probabilities by repeating the random experiment many times and counting how many times each pattern appears first. So if the random experiment in the case of the given Google interview question is repeated 1000000 times and the pattern HHT appears first in 665000 of these experiments (and the pattern HTT appears first in 33500 of the experiments), then the estimated values of the probabilities are 0.665 (for HHT) and 0.335 (for HTH).

The program receives as command line parameters the two patterns (of H-s and T-s) and optionally the number of trials (i.e. times to repeat the random experiment) and the number of threads that will be used to solve the problem. If the optional arguments are not provided then the default number of trials is 1000000 and the default number of threads is 10.

A sample execution (if you are using Java) may be:

```
java ProbabilityEstimation HHT HTT 2000000 8
```

Or if you are using POSIX threads may be:

```
./ProbabilityEstimation HHT HTT 2000000 8
```

The expected result will look like:

```
Estimated values with 2000000 random experiments with 8 threads:
The probability that HHT appears first is 0.66671
The probability that HTT appears first is 0.33329
```

Another sample execution (if you are using Java) may be:

```
java ProbabilityEstimation THT HTH
```

Or if you are using POSIX threads may be:

```
./ProbabilityEstimation THT HTH
```

The expected result will look like:

```
Estimated values with 1000000 random experiments with 10 threads:
The probability that HTH appears first is 0.49998
The probability that THT appears first is 0.50002
```

a. Solve the given problem using Java multithreading.
b. Solve the given problem using POSIX threads.

NOTE: your solution may be tested with patterns of different lengths, for example HTT and TH, but it will not be tested with patterns where one is suffix of another for example HTT and TT, in order to avoid ambiguity (both patterns appearing in the same time).

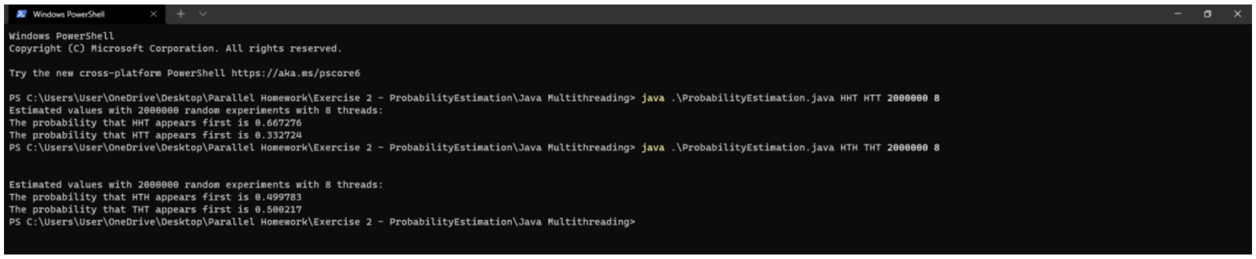
Edison Ponari
CEN 330 / CEN A

Solution:

Source code is provided in "Exercise 2 – ProbabilityEstimation" folder. I wrote a program that will take two patterns containing H(Head) and T(Tail) and calculate the probability of one of them showing up before the other one. I set up 2 global variables to count how many times a pattern has showed up first out of n tries. Each thread will do work only when currentTry%threadCount == currentThread. This makes sure that the workload is balanced on all threads with a maximum difference of 1. Until the thread hasn't finished its work, the generated pattern inside the thread is concatenated randomly with a 'H' or 'T'. When the generated pattern ends with one of the patterns we are searching, we add 1 to the local count for that pattern in the thread and empty the pattern generated since we need to start the experiment from the beginning. After the work is done, the thread has a synchronized region where the local counts are added to the global count for each pattern. The POSIX C program follows the same logic but I use mutex to synchronize the region and create my own functions for strings since C doesn't have libraries as rich as Java.

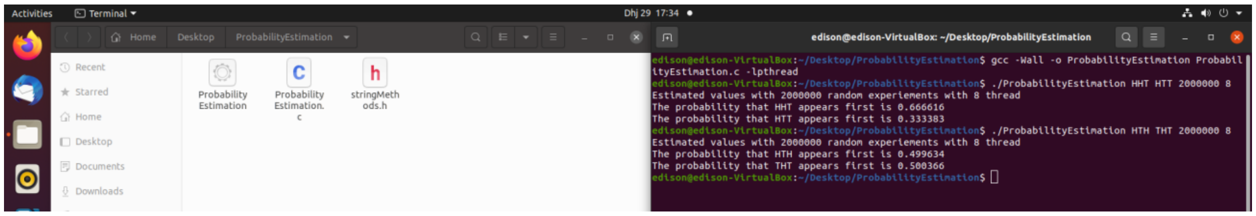
The source code is well documented for each version of the exercise with comments.

Java Multithreading:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 2 - ProbabilityEstimation\Java Multithreading> java .\ProbabilityEstimation.java HHT HTT 2000000 8
Estimated values with 2000000 random experiments with 8 threads:
The probability that HHT appears first is 0.667276
The probability that HTT appears first is 0.332724
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 2 - ProbabilityEstimation\Java Multithreading> java .\ProbabilityEstimation.java HTH THT 2000000 8


Estimated values with 2000000 random experiments with 8 threads:
The probability that HTH appears first is 0.499783
The probability that THT appears first is 0.500217
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 2 - ProbabilityEstimation\Java Multithreading>
```

POSIX Threads:



```
edison@edison-VirtualBox:~/Desktop/ProbabilityEstimation$ gcc -Wall -o ProbabilityEstimation Probabil
ityEstimation.c -lpthread
edison@edison-VirtualBox:~/Desktop/ProbabilityEstimation$ ./ProbabilityEstimation HHT HTT 2000000 8
Estimated values with 2000000 random experiements with 8 thread
The probability that HHT appears first is 0.666616
The probability that HTT appears first is 0.333383
edison@edison-VirtualBox:~/Desktop/ProbabilityEstimation$ ./ProbabilityEstimation HTH THT 2000000 8
Estimated values with 2000000 random experiements with 8 thread
The probability that HTH appears first is 0.499634
The probability that THT appears first is 0.500366
edison@edison-VirtualBox:~/Desktop/ProbabilityEstimation$ 
```

PROBLEM 3 (25 pts) - Random restart hill climbing

Hill climbing is a local search algorithm used in optimization problems (to determine an approximate maximum value of a function, which is known as the objective function). It is an iterative algorithm that starts with an arbitrary point (which may be given or may be generated randomly), then attempts to find a better solution by making an incremental change to the solution. If the change produces a better solution, another incremental change is made to the new solution, and so on until no further improvements can be found. The major drawback of this algorithm is that it may converge on a local maximum (thus providing a suboptimal solution). In order to avoid this drawback there is another version of this algorithm known as: random restart hill climbing.

In random restart hill climbing, the classical hill climbing is applied several times and among the found answers, the best one is picked as the final answer.

In this problem you are required to apply parallel random restart hill climbing in order to find the maximal value over a specified rectangular region for a two variable function, which will be already hard-coded in your source file (with signature):

double objectiveFunction(double x, double y).

The program will read as input the minX, maxX, minY, maxY as the bounds of the rectangular region and the stepSize parameter. So, starting at a random initial point (x_0, y_0), it will check the eight "neighboring" points: $(x_0 - stepSize, y_0)$, $(x_0 + stepSize, y_0)$, $(x_0, y_0 - stepSize)$, $(x_0, y_0 + stepSize)$, $(x_0 - stepSize, y_0 - stepSize)$, $(x_0 + stepSize, y_0 - stepSize)$, $(x_0 - stepSize, y_0 + stepSize)$ and $(x_0 + stepSize, y_0 + stepSize)$. The "neighboring" point which provides the largest value of the objective function is found and (if this value is larger than the current value), the "best neighbor" will become our current node and the procedure is repeated.

a. Write a program using Java multi-threading which reads as input the total number of times that hill climbing will be repeated, the number of threads that will be created, the values minX, maxX, minY, maxY and the stepSize. Finally it will print as result the largest value of the objective function (that is found) and the values x and y of the point for which it is obtained.

b. Write a program using POSIX threads which reads as input the total number of times that hill climbing will be repeated, the number of threads that will be created, the values minX, maxX, minY, maxY and the stepSize. Finally it will print as result the largest value of the objective function (that is found) and the values x and y of the point for which it is obtained.

Note: For testing purposes you may use the objective function defined as:

```
double objectiveFunction(double x, double y){
     return 50 + 20*x*y - 3x*x*y -  2*x*y*y;
}
```

And minx =0, maxX = 10, minY = 0, minY = 10, stepSize=0.01, so the final answer is expected to be approximately 99.4 obtained for x = 2.23 and y = 3.33

Edison Ponari
CEN 330 / CEN A

Solution:

Source code is provided in "Exercise 3 - Random Restart Hill Climbing" folder. I wrote a program that has a certain 2 variable (x, y) function and given the number of times to repeat the algorithm, number of threads to use, minimum x, maximum x, minimum y, maximum y and the step size to use, calculates the maximum value of this function in that region and the x and y to achieve that maximum value. Since the point of doing it in parallel is to start the algorithm from different point, there were 2 approaches I could take:

1) start the algorithm from random points.

2) start the algorithm from each square unit in the given limits.

I went with the second approach since it made the algorithm way faster and the number of tries was set a maximum repeating step to find the local maximum. Each thread has a set of squares to search the maximum from and communicate about the found maximum with global variables. Example (1,1) to (10,10) with 10 threads (0 to 9):

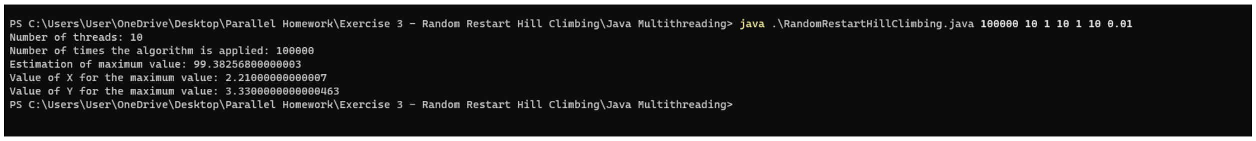
9,9

2	3	4	5	6	7	8	9	0
3	4	5	6	7	8	9	0	1
4	5	6	7	8	9	0	1	2
5	6	7	8	9	0	1	2	3
6	7	8	9	0	1	2	3	4
7	8	9	0	1	2	3	4	5
8	9	0	1	2	3	4	5	6
9	0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7	8

1,1

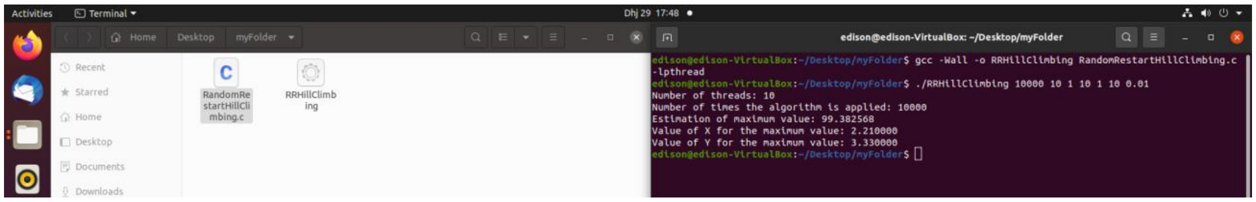
The source code is well documented for each version of the exercise with comments.

Java Multithreading:



```
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 3 - Random Restart Hill Climbing\Java Multithreading> java .\RandomRestartHillClimbing.java 100000 10 1 10 1 10 0.01
Number of threads: 10
Number of times the algorithm is applied: 100000
Estimation of maximum value: 99.38256800000003
Value of X for the maximum value: 2.21000000000007
Value of Y for the maximum value: 3.3300000000000463
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 3 - Random Restart Hill Climbing\Java Multithreading>
```

POSIX Thread:



```
edison@edison-VirtualBox:~/Desktop/myFolder$ gcc -Wall -o RRHillClimbing RandomRestartHillClimbing.c
-lpthread
edison@edison-VirtualBox:~/Desktop/myFolder$ ./RRHillClimbing 10000 10 1 10 1 10 0.01
Number of threads: 10
Number of times the algorithm is applied: 10000
Estimation of maximum value: 99.382568
Value of X for the maximum value: 2.210000
Value of Y for the maximum value: 3.330000
edison@edison-VirtualBox:~/Desktop/myFolder$ 
```

PROBLEM 4 (25 pts) - Multithreaded client-server application

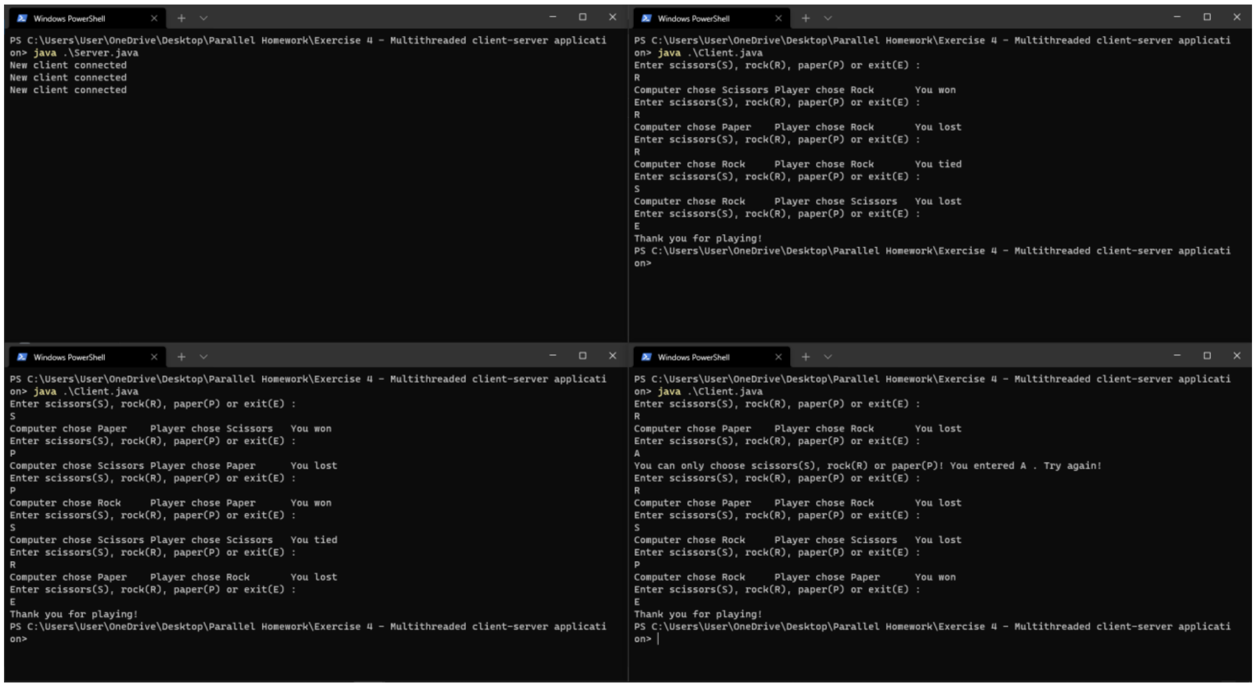
Multi-threading in Java is frequently used in client/server applications (using sockets) while programming the server side of the application. Design a program of the client/server model where the server serves several clients simultaneously handling each client in a separate thread. For example, this may be a chatting program, a game etc.

Solution:

Source code is provided in "Exercise 4 - Multithreaded client-server application" folder. I wrote a program that has a Server class using TCP protocol and a Client class to send request to the server. For each socket the server opens to communicate with a client, it creates a new object that inherits from the Thread class and overrides the run method. The run method contains a game of Rock Paper Scissors to be played against a computer that draws one of the 3 choices in random. I had to look up most of the socket programming part since there were a lot of errors related to how the input/output is sent/read by the server and the user. I tried to create a queue with user to wait to pair them with each other to play the game between them but sending messages to each other through the server was not working like I wanted to so I stuck to this solution to play against the computer.

The source code is well documented with comments.

Java Multithreading (3 players and 1 server):



```
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 4 - Multithreaded client-server applicati
on> java .\Server.java
New client connected
New client connected
New client connected
```

```
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 4 - Multithreaded client-server applicati
on> java .\Client.java
Enter scissors(S), rock(R), paper(P) or exit(E) :
R
Computer chose Scissors Player chose Rock       You won
Enter scissors(S), rock(R), paper(P) or exit(E) :
R
Computer chose Paper    Player chose Rock       You lost
Enter scissors(S), rock(R), paper(P) or exit(E) :
R
Computer chose Rock     Player chose Rock       You tied
Enter scissors(S), rock(R), paper(P) or exit(E) :
S
Computer chose Rock     Player chose Scissors   You lost
Enter scissors(S), rock(R), paper(P) or exit(E) :
E
Thank you for playing!
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 4 - Multithreaded client-server applicati
on>
```

```
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 4 - Multithreaded client-server applicati
on> java .\Client.java
Enter scissors(S), rock(R), paper(P) or exit(E) :
S
Computer chose Paper    Player chose Scissors   You won
Enter scissors(S), rock(R), paper(P) or exit(E) :
P
Computer chose Scissors Player chose Paper      You lost
Enter scissors(S), rock(R), paper(P) or exit(E) :
P
Computer chose Rock     Player chose Paper      You won
Enter scissors(S), rock(R), paper(P) or exit(E) :
S
Computer chose Scissors Player chose Scissors   You tied
Enter scissors(S), rock(R), paper(P) or exit(E) :
R
Computer chose Paper    Player chose Rock       You lost
Enter scissors(S), rock(R), paper(P) or exit(E) :
E
Thank you for playing!
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 4 - Multithreaded client-server applicati
on>
```

```
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 4 - Multithreaded client-server applicati
on> java .\Client.java
Enter scissors(S), rock(R), paper(P) or exit(E) :
R
Computer chose Paper    Player chose Rock       You lost
Enter scissors(S), rock(R), paper(P) or exit(E) :
A
You can only choose scissors(S), rock(R) or paper(P)! You entered A . Try again!
Enter scissors(S), rock(R), paper(P) or exit(E) :
R
Computer chose Paper    Player chose Rock       You lost
Enter scissors(S), rock(R), paper(P) or exit(E) :
S
Computer chose Rock     Player chose Scissors   You lost
Enter scissors(S), rock(R), paper(P) or exit(E) :
P
Computer chose Rock     Player chose Paper      You won
Enter scissors(S), rock(R), paper(P) or exit(E) :
E
Thank you for playing!
PS C:\Users\User\OneDrive\Desktop\Parallel Homework\Exercise 4 - Multithreaded client-server applicati
on> |
```

Edison Ponari
CEN 330 / CEN A