Software Testing Assignment (CSE 4th Year)

Rules:

1. You can take help from friends and experts.
2. Use of AI tools like ChatGPT, Copilot and others is legal.
3. In case of 1 and 2 please mention source of your answer.
4. This assignment contains three parts.
5. Read file named "index.html" to have clear idea of API.
6. Submission Date: 24/Nov/2025

Problem:

Indian developers are using API named **"speed"** to analyze algorithms (for API Details read index.html) from past 5 years. This API was designed to run sorting algorithms (by default selection sort) in normal (i.e. sequential) and chunk (i.e. parallel) mode.

In **normal model API** works as follows:

(a) Take entire array as single chunk.
(b) Sort the array in one go.
(c) Display the processing or sorting time in milliseconds.

However, in **parallel mode API** do following:

(a) Divide given array into chunks.
(b) Parallelly sort each chunk using default selection sort algorithm.
(c) Merge sorted chunks into one big, sorted chunk using merge procedure.
(d) Display algorithm's computation or processing time in milliseconds.

Part 1 (5 Marks):

Suppose during tariff war friend of Pakistan **Mr. Doland Trump** imposed **100% tariff** on the use of API. Now being smart Indian Software Tester and Developer, you aimed to do reverse engineering to re-write the entire API just by reading the available description (see index.html).

However, before re-writing entire API (which is aim of part 1) you need to understand it's working and analyze it. To achieve this objective, you designed a procedure (plan) as discussed in part 2 and part 3. In part 2 you decided to re-write two files and analyze it. In part 3 you decided to add new features to existing API. After this you decided to come back to Part 1 and re-write the complete API.

Part 2 (5 Marks):

(1) Run file named **"SpeedUni"** using command **"java SpeedUni"** using inputs as shown in figure Input1. This file runs algorithm in **"normal mode".** Note your runtime. **[0.5 Marks]**

(2) Run file named **"SpeedMulti"** using command **"java SpeedMulti"** using inputs as shown in figure Input2. This file runs algorithm in **"parallel mode".** This file divides given input into two chunks, parallelly sort each chunk, merge both chunks to get one big, sorted chunk. Note your runtime. **[0.5 Marks]**

(3) Note your observations by re-running **"SpeedUni"** and **"SpeedMulti"** on different and same inputs. **[0.5 Marks]**

(4) Read API (using file index.html) and then write source code of files **"SpeedUni.java"** and **"SpeedMulti.java"** & store them in files named **"MySpeedUni.java" and "MySpeedMulti.java"** respectively. Once source code is written then run your code and compare your results with results obtained in (1), (2) and (3). **[3.5 Marks]**

Part 3 (5 Marks):

(1) By default, **selection sort** is used as prime sorting technique in the given API. Upgrade the API by adding **Bubble Sort** in the core API. **[3.0 Marks]**

(2) Write source code of files named "**SpeedUni"** & "**SpeedMulti"** and store them in files **"MyBubbleUni.java"** & **"MyBubbleMulti.java"** respectively. These files uses bubble sort as primary sorting algorithm instead of default selection sort. **[1.0 Marks]**

(3) Run your code as done in part 1 (1), (2) and (3) and compare your results to that obtained in part 1. Also describe differences in results with proper reason. **[1.0 Marks]**

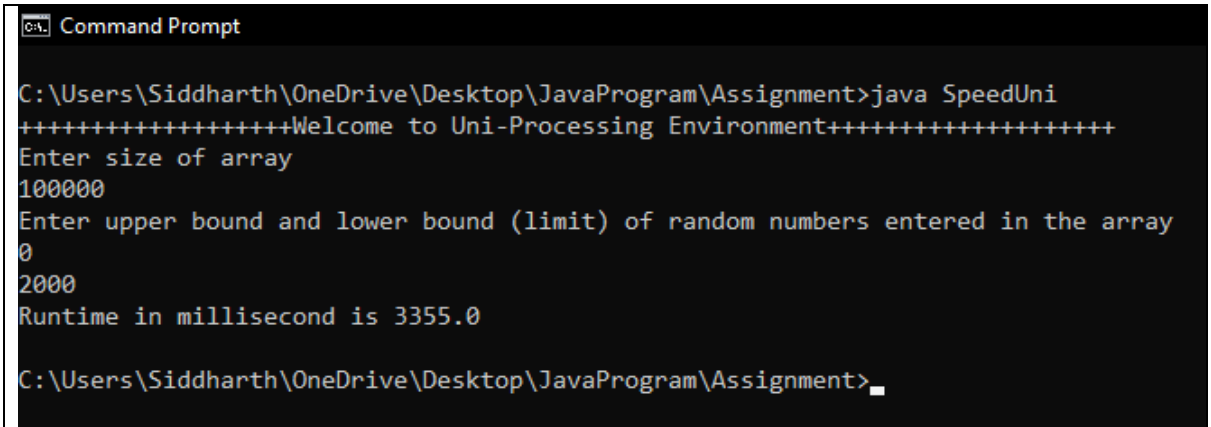

Figure: Input1

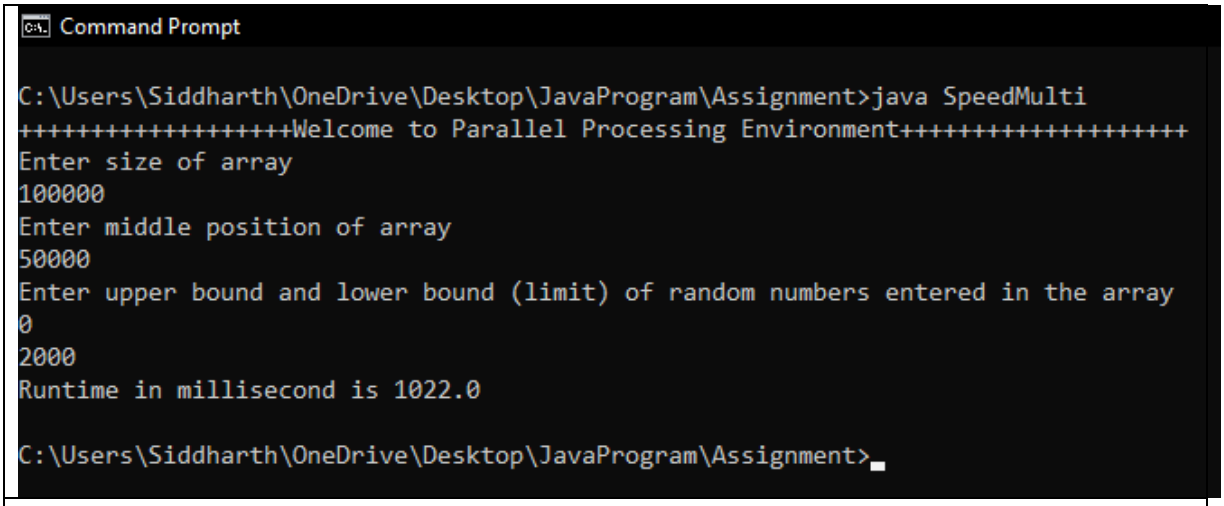

Figure: Input2

Note: Figure 1 & 2 results are obtained by running API on Intel i3, 6th Generation, Windows 10 system.