BACS1014 PROBLEM SOLVING AND PROGRAMMING
ASSIGNMENT

1.0 General Information

Objective:	Apply the programming concepts and skills that you learned to write a simple program.
Task Summary:	Design and develop a Running Man simulation program using random number generation.
Assessment Weight:	**50 % of Coursework** component.
Team Size:	2 (**Team** assignment)
Submission Mode:	Softcopy (refer to Section **5.0 Report**).
Submission Date:	Friday of Week 12

2.0 Brief Description / Purpose

Imagine a long-distance race between two persons through a hilly area and uneven terrain. Rain, mud and rocks litter the trail path that both runners traverse on. As you can visualise, it is not a smooth run for both runners where they may pick up speed at certain flat stretches but slow down to a crawl at rocky paths. And each runner will have their unique run experience to share at the end of the race.

You have been hired to develop a **program to simulate the above running scenario for runners using random number generation.**

3.0 Scenario & Program Specifications

The runners begin the race at Square 1 from a total square count of 60. Each square represents a position along the race course, and the finish line is at Square 60. The first runner to reach or pass Square 60 is adjudged to have won the race. Occasionally the runner loses pace when encountering bad race conditions.

The race has a timer where it ticks once per second, i.e. frequency of 1 Hz. With each tick, your program must adjust the position of the runners according to the following rules in Table 1: -

Runner #	Movement	% of time	Actual move
1	Sprint	50	3 squares upwards
	Jog	20	1 square upwards
	Walk	30	6 squares downwards
2	Fast Sprint	30	5 squares upwards
	Run	20	3 squares upwards
	Walk	10	2 squares downwards
	Crawl	20	4 squares downwards
	Sleep	20	No movement

Table 1

Your program should be developed in stages (following (a), (b), and (c) below). Test one part and ensure it is working before adding in the requirements of another part. This makes your program easier to manage and to locate the source of errors. Submit only ONE final version.

(a) -- Use variables to keep track of the positions of the runners. Position numbers are 1 - 60 mirroring the square positions.

-- Each runner starts the race at Position 1, considered to be the starting gate. If after generating a random number causes the runner to 'walk' or 'crawl' before even departing from the starting gate, move the runner back to Position 1.

-- Generate the percentages in Table 1 by producing a random integer, *i*, in the range $1 \le i \le 10$. For example, for Runner #1, perform a Sprint when $1 \le i \le 5$, a Jog when $6 \le i \le 7$, or a Walk when $8 \le i \le 10$. Use a similar technique to move Runner #2.

-- Begin the race with the printout

BANG!!!

AND AWAY THEY GO…!

(b) -- For each tick of the clock (i.e. each repetition of a loop), print a 60-position line showing the letters '1' and '2' to represent the positions of the runners.

-- Occasionally, the runners will land on the same square. When that happens, your program should print 'GOTCHA!' beginning at that position. All other positions should be blank, other than the '1', '2', and 'GOTCHA!'.

(c) -- After each line is printed, test if either runner has reached or passed Square 60. If so, then print the winner and terminate the simulation. If Runner #1 wins, print 'RUNNNER #1 WINS!'. If Runner #2 wins, print 'RUNNER #2 WINS!'. If both runners win on the same tick of the clock, print 'IT'S A TIE!'. If neither runner wins, perform the loop again the to simulate the next tick of the clock (in other words, the race continues).

4.0 Program Testing & Outputs

Run your final program using different sets of test data. Show your planned **test data and expected output** for *at least 3 runs*. Present the 3 runs in a table and the table must be included in your report.

You should cover as many different scenarios as possible (e.g. random numbers generated are integers, when a runner crosses the end line, when both runners meet on the same square, etc). For each run, capture the screenshots to be included in your report. **Make sure the actual output of your program runs mirror exactly the test data and expected output.**

5.0 Report

Your **Assignment Report** should include:

- **Front Cover** TAR UC Name, Faculty Name, Course Code & Title, Assignment Title, Student Name, and Programme.
- **Declaration of Originality –** to be included in your report (Refer to Section 7.0).
- **Table of Content –** list of contents and their page numbers.
- **Brief Description / Purpose** of the program (Refer to Sections 2.0 & 3.0).
- **Overall Program Design** *Flow Chart* showing the structure of the whole program.
    - Your flowchart should NOT be a line-for-line version of your program. It should show the general steps in solving the problem.
- **Added features** Describe what extra features you have included in the system. Explain why you think they are useful and also the motivation behind them.
- **Program Testing & Outputs** Tables of test data & expected outputs each followed by screenshots for the runs that you have planned (Refer to Section 4.0). There should be a minimum of 3 sets; each set should start on a new page.
    - Your screenshots should show what is displayed for the whole run.
    - Note on screen output
        - text should be displayed in black on a white background
        - remember to include those that show your extra features

- **Constants & variables** Include in your report the **definitions** / **declarations** of constants and variables from your program. Then, using 2 tables (one for constants and one for variables), list and describe each constant (value, data type, purpose) and variable (data type & purpose).

- **Program Listing** Include the source code **from Visual Studio** in your report.
  - Your program should be indented appropriately and have useful comments for the different parts or sections.

- The report should include a proper analysis of the program as well as discussions and an overall conclusion.

- The report should be tidily formatted and submitted as one entity. Insert the signed ***Declaration of Originality*** after the Cover Page.

6.0 Assessment Criteria

This assignment contributes 50% to the overall coursework, but is graded to a full 100% for ease of computation. The allocation of marks is shown below.

Areas	Marks Allocated (%)
a. Program design	20
b. Fulfilling the requirements of the task	20
c. Program efficiency & readability	20
d. Additional features	10
e. Report	30
Total	**100**

<u>Marks will be deducted:</u>

- if instructions are not followed, e.g: no soft copy, screenshots color unsuitable, incomplete report contents, etc.
- if you are found to have plagiarized work (refer to Section 7.0)

7.0 Ethics and Punctual Submission

7.1 Plagiarism and Collusion

All submitted work must be **original,** and not previously submitted for any assessment, nor outsourced to any party. Students found to have copied others' work, or used them without properly acknowledging the sources, will be penalized for plagiarism. The **Declaration for Originality** of work (refer below) must be **signed** and submitted with the Report.

It is not acceptable for a student to submit program(s) that are similar or identical to those of other students by simply and cosmetically disguising it with some modifications. In such a case, **both the students (who copied and who allowed others to copy) will be penalized**.

Declaration of Originality

I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own work. I understand that I will be penalized if I have not complied with TAR UC's Plagiarism policy.

Signature: _______________ Name: ______________________ Date: __________

7.2 Late Submission

This assignment should be submitted by the due date as stated unless as revised or approved by your lecturer. Marks will be deducted for late submission at **10%** per day late, unless approval is granted **before** the due date.