##### T:\NP 9 Partnership & Branding\2. Branding\1. Logos and collaterals\College logo_jpeg and png files\Flame\Black and white\YIJC_Flame_Black.pngYISHUN INNOVA JUNIOR COLLEGE

JC 2 MID YEAR EXAMINATION

**Higher 2**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CANDIDATE  NAME | |  | |  | | | | |
| **skrrt** | | | | |
|  | | | | |
| CG | |  | |  | |  | INDEX NO |  |  | |
| 6969 | | | 12345 |  | |
|  | |  |  |  | |

|  |  |
| --- | --- |
| **COMPUTING**  Paper 1 Practical Paper | **9597/01**  **29 August 2019**  **3 hour 15 minutes**  **100 Marks** |

Additional Materials:

Removable storage device with the following files:

* EVIDENCE-DOC.doc
* HEIGHTDATA.TXT
* 1000WORDS.TXT
* PSEUDOCODE\_TASK\_3\_3.TXT

## READ THESE INSTRUCTIONS FIRST

Type in the EVIDENCE-DOC document the following:

* Candidate details
* Programming language used

Answer **all** questions.

All tasks must be done in the computer laboratory. You are not allowed to bring in or take out any pieces of work or materials on paper or electronic media or in any other form.

Approved calculators are allowed.

The number of marks is given in brackets [ ] at the end of each question or part question.

Copy and paste required evidence of program codes using ‘Courier New’ font and screenshots of the outputs into the EVIDENCE-DOC document. **Submit your EVIDENCE-DOC in the thumb drive provided.**

**Task 1.1:**

Write program code to:

* read the entire contents of HEIGHTDATA.TXT.
* determine if the boy’s height lies within the normal range.
* display the contents using the format given below

**Evidence 1.1:**

Your program code. [6]

**Evidence 1.2:**

Screenshot of the output. [2]

**Task 1.2:**

Write program code to:

* determine the correct height for those entries outside the normal range.
* display the amended contents using the format given below.

**Evidence 1.3:**

Your program code. [6]

**Evidence 1.4:**

Screenshot of the output. [1]

**Evidence 2.1:**

Describe the binary search algorithm. [2]

**Task 2.1:**

Write a program BinarySearch(lst, item) to search for an item in the list lst using the binary search algorithm.

The program will:

* import the sorted list of words, given in the file 1000WORDS.TXT, into a simple array dataset.
* report whether or not the item is found in the list. If found, output the index position and the list of words examined by the program during the binary search.

**Evidence 2.2:**

Your program code and the screenshot for the following searches:

* BinarySearch(dataset, "WORD")
* BinarySearch(dataset, "WORDA")
* BinarySearch(dataset, "TRADE") [8]

**Task 2.2:**

Modify the code BinarySearch(lst, item) written in Task 2.1.

Your program will:

1. perform a partial search for the word in the list lst starting with the given letter(s), item
2. perform a linear search near the index found in step (1) to return a list of words starting with the given letter(s)

**Evidence 2.3:**

Your program code and the screenshot for the following searches:

* BinarySearch(dataset, "TR")
* BinarySearch(dataset, "RE") [5]

**Task 3.1**

Write a program code to define the Node and ExpressionTree classes.

**Evidence 3.1**

Your program code for Task 3.1. [12]

**Task 3.2**

Write a function IsOperator(s) that takes in a string s, and returns True if it is a standard arithmetic operator and returns False if otherwise.

**Evidence 3.2**

Your program code for Task 3.2. [2]

**Task 3.3**

Write a code to implement the Insert method for the ExpressionTree class from this pseudocode.

You may use the text file PSEUDOCODE\_TASK\_3\_3.TXT as a basis for writing your code.

**Evidence 3.3**

Your program code for Task 3.3. [7]

**Task 3.4:**

Write a code for the Display method for the ExpressionTree class which displays the contents of Tree in index order.

**Evidence 3.4**

Your program code for Task 3.4. [4]

**Task 3.5**

Write a sequence of program statements to:

* create an expression tree
* add the data items based on the sequence of commands given
* display the array contents

**Evidence 3.5**

Your program code for Task 3.5. [3]

**Evidence 3.6**

Screenshot showing the output from running the program in Task 3.5. [1]

**Task 3.6**

The infix notation can be obtained by performing an in-order traversal in the expression tree. Write a code for the infix method for the ExpressionTree class to generate the infix notation for a complete expression tree.

**Evidence 3.7**

Your program code for Task 3.6. [6]

**Evidence 3.8**

Screenshot showing the output from running the program in Task 3.6. [1]

**Task 3.7**

Write a code for the calculate method to evaluate and return the numerical answer for the expression, rounded off to 2 decimal places.

**Evidence 3.9**

Your program code for Task 3.7. [3]

**Evidence 3.10**

Screenshot showing the output from running the program in Task 3.7. [1]

**Task 4.1:**

Write a program code to generate and display an empty square grid of size *n*, ie *n* rows by *n* columns. The minesweeper grid for *n* = 5 is as shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Your code should use a suitable data structure and fixed loop(s) to display the grid.

**Evidence 4.1:**

Your program code and screenshot of an empty grid of size 5. [3]

**Task 4.2:**

Write a program code to randomly place a bomb, represented by "X", within the grid. Populate all the neighbouring cells by increasing their values to 1 to indicate the presence of this one bomb in the neighbourhood.

**Evidence 4.2:**

Your program code and two different screenshots of the grid (*n*=5). [5]

**Task 4.3:**

Modify the code written in Task 4.2 to randomly place two bombs within the grid. Populate all the neighbouring cells with the correct values to indicate the presence of the bombs in the neighbourhood.

**Evidence 4.3:**

Your program code and the screenshot of the grid (*n*=5) with 2 bombs. [4]

**Task 4.4:**

Modify the code written in Task 4.3 to generate *k* numbers of bombs within a grid of size *n* and correctly display all the values in the neighbouring cells surrounding the bombs.

**Evidence 4.4:**

Your program code and the screenshots of the minesweeper grids for the following levels of difficulty.

* Beginner (grid size *n*=5; no. of bombs *k*=3)
* Intermediate (grid size *n*=6; no. of bombs *k*=8)
* Expert (grid size *n*=8; no. of bombs *k*=20) [8]

**Task 4.5:**

Write a program code to play the minesweeper game. Your code will:

* prompt the player to select the level of difficulty
* generate the Minesweeper grid
* display a “blank” grid with ‘**-**’ for each of the cell
* prompt the player to input the coordinates of a cell he wishes to open
  + If the opened cell is a bomb (“X”), declare “Game Over!”, show the grid and display the player’s score.
  + If the opened cell is not a bomb, show the updated grid with the opened cell, increase the player’s score by 1 and continue with the game.
* declare “You have Won!” when the player has opened all the possible cells and display the player’s score.

**Evidence 4.5:**

Your program code and a screenshot of a game. [10]