2.7 Dictionaries

Name: _____

Class: _____

Date: _____

Time: **70 minutes**

Marks: **37 marks**

Comments:

Q1.

(a) This question refers to the subroutine `CreateTileDictionary`.

The points values for the letters J and X are to be changed so that they are not worth as many points as the letters Z and Q.

Adapt the subroutine `CreateTileDictionary` so that the letters J and X are worth 4 points instead of 5 points.

Test that the changes you have made work:
- run the **Skeleton Program**
- enter `2` at the main menu
- enter `1` to view the letter values.

Evidence that you need to provide

(i) Your PROGRAM SOURCE CODE for the amended subroutine `CreateTileDictionary`.

(1)

(ii) SCREEN CAPTURE(S) showing the results of the requested test.

(1)

(b) This question refers to the subroutine `Main`.

Currently each player starts with 15 letter tiles. In the `Main` subroutine `StartHandSize` is set to a value of 15.

Change the subroutine `Main` so that the user can choose what the value for `StartHandSize` will be.

Before the main menu is displayed and before the first iteration structure in `Main` the message "`Enter start hand size: `" should be displayed and the user's input should be stored in `StartHandSize`. This should happen repeatedly until the user enters a value for the start hand size that is between 1 and 20 inclusive.

Test that the changes you have made work:

- run the **Skeleton Program**
- enter `0` when asked to enter the start hand size
- enter `21` when asked to enter the start hand size
- enter `5` when asked to enter the start hand size
- enter `1` at the main menu.

Evidence that you need to provide

(i) Your PROGRAM SOURCE CODE for the amended subroutine `Main`.

(4)

(ii) SCREEN CAPTURE(S) showing the requested test. You must make sure that evidence for all parts of the requested test is provided in the SCREEN CAPTURE(S).

(1)

(c) This question refers to the subroutine `CheckWordIsValid`.

When a player enters a word, a linear search algorithm is used to check to see if the word entered is in the list of `AllowedWords`. The subroutine `CheckWordIsValid` is to be changed so that it uses a more time-efficient search algorithm.

Change the `CheckWordIsValid` subroutine so that it uses a binary search algorithm instead of a linear search algorithm.

You **must write your own search routine** and not use any built-in search function that might be available in the programming language you are using.

Each item in `AllowedWords` that is compared to the word that the user entered should be displayed on the screen.

Figure 2 shows examples of how the new version of `CheckWordIsValid` should work if `AllowedWords` contained the items shown in **Figure 1**.

Figure 1

BIG
BUG
FED
GET
JET
NOT
SIP
WON

Figure 2

1) If the user enters the word BIG then a value of `True` should be returned and the words GET, BUG, BIG should be displayed, in that order.
2) If the user enters the word JET then a value of `True` should be returned and the words GET, NOT, JET should be displayed, in that order.
3) If the user enters the word ZOO then a value of `False` should be returned and the words GET, NOT, SIP, WON should be displayed, in that order.

Test that the changes you have made work:

- run the **Skeleton Program**
- if you have answered part (b), enter `15` when asked to enter the start hand size; if you have not answered part (b) yet then skip this step
- enter `2` at the main menu
- enter the word `jars`.

Evidence that you need to provide

(i) Your PROGRAM SOURCE CODE for the amended subroutine `CheckWordIsValid`.

(8)

(ii) SCREEN CAPTURE(S) showing the requested test.

(1)

(d) This question extends the functionality of the game.

After spelling a valid word the player decides which one of four options to select to determine how many tiles will be added to their hand. Before choosing they can look at the values of the tiles.

It would help the player make their decision if they were aware of how useful each letter was by knowing the frequency with which each letter appears in the list of allowed words.

The program is to be extended so that when the player chooses to view the tile values they are also shown the number of times that each letter appears in the list of allowed words.

What you need to do

Task 1
Create a new subroutine called `CalculateFrequencies` that looks through the list of allowed words and displays each of the 26 letters in the alphabet along with the number of times that the letter appears in the list of allowed words, which the subroutine has calculated.

Task 2

Modify the `DisplayTileValues` subroutine so that after displaying the tile values it also calls the `CalculateFrequencies` subroutine.

Task 3
Test that the changes you have made work:

- run the **Skeleton Program**
- if you have answered part (b), enter `15` when asked to enter the start hand size; if you have not answered part (b) yet then skip this step
- enter `2` at the main menu
- enter `1` to display the letter values.

Evidence that you need to provide

(i) Your PROGRAM SOURCE CODE for the new subroutine `CalculateFrequencies`, the amended subroutine `DisplayTileValues` and any other subroutines you have modified when answering this question.

(8)

(ii) SCREEN CAPTURE(S) showing the requested text.

(1)

(e) The scoring system for the game is to be changed so that if a player spells a valid word they score points for all valid words that are a prefix of the word entered. A prefix is the first **x** characters of a word, where **x** is a whole number between one and the number of characters in the word.

In the **Skeleton Program**, `AllowedWords` contains the list of valid words that have been read in from the **Data File aqawords.txt**.

Example

If the user enters the word TOO they will be awarded points for the valid words TOO and TO as TO is a prefix of the word TOO. They will not be awarded points for the word OO even though it is a valid word and is a substring of TOO because it is not a prefix of TOO.

Example
If the user enters the word BETTER they will be awarded points for the words BETTER, BET and BE as these are all valid prefixes of the word entered by the user. They would not be awarded points for BETT or BETTE as these are not valid English words. They would not be awarded points for BEER as even though it is contained in the word BETTER it is not a prefix.

Example
If the user enters the word BIOGASSES they will be awarded points for the words BIOGASSES, BIOGAS, BIOG, BIO and BI as these are all valid prefixes of the word entered by the user. They would not be awarded points for BIOGA, BIOGASS or BIOGASSE as these are not valid English words. They would not be awarded points for GAS as even though it is contained in the word BIOGASSES it is not a prefix.

Example
If the user enters the word CALMIEST they will not be awarded any points as even though CALM at the start is a valid word the original word entered by the user, CALMIEST, is not.

Example
If the user enters the word AN they will be awarded points for the word AN. They would not be awarded points for A even though A at the start is a valid word as points are only awarded for words that are at least two letters long.

What you need to do

Task 1
Write a **recursive** subroutine called `GetScoreForWordAndPrefix` that, if given a valid word, returns the score of the word added to the score for any valid words that are prefixes of the word.

To get full marks for this task the `GetScoreForWordAndPrefix` subroutine must make use of recursion in an appropriate way to calculate the score for any prefixes that are also valid words.

If your solution uses an alternative method to recursion you will be able to get most but not all of the available marks for this question.

Task 2
Modify the `UpdateAfterAllowedWord` subroutine so that it calls the new `GetScoreForWordAndPrefix` subroutine instead of the `GetScoreForWord` subroutine.

Task 3
Test that the changes you have made to the program work:

- run the **Skeleton Program**
- if you have answered part (b), enter `15` when asked to enter the start hand size; if you have not answered part (b) yet then skip this step
- enter `2` at the main menu
- enter the word `abandon`
- enter `4` so that no tiles are replaced.

Evidence that you need to provide

(i) Your PROGRAM SOURCE CODE for the new subroutine `GetScoreForWordAndPrefix`, the amended subroutine `UpdateAfterAllowedWord` and any other subroutines you have modified when answering this question.

(11)

(ii) SCREEN CAPTURE(S) showing the results of the requested test.

(1)

(Total 37 marks)