ST ANDREW'S JUNIOR COLLEGE

H2 COMPUTING **9597**

JC2 PRELIMINARY EXAMINATIONS **19 SEP 2013**

TIME: 0900 – 1215 hrs **3 hours 15 mins**

Additional Materials: Pre-printed A4 Paper
Removable storage device
Electronic version of `TOP_TEAM.txt` data file
Electronic version of `SPORT+TEAM.txt` text file
Electronic version of `TEAMS.txt` data file
Electronic version of `RESULTS.txt` data file
Electronic version of `MORSE.txt` text file
Electronic version of EVIDENCE-DOC document

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.

All tasks and required evidence are numbered.

The number of marks is given in brackets [ ] at the end of each task.

Copy and paste required evidence of program listing and screen shots into the EVIDENCE-DOC document.

At the end of the examination, print out your EVIDENCE-DOC and fasten your printed copy securely together.

Total marks for this paper is **100** marks.

This question paper consists of 6 printed pages.

1 A program is to process the total points of football league teams based on number of wins, draws and losses. The program can be run every day.

The program reads from file TOP_TEAM.txt the current highest points and team name from running the program on previous days.
The program specification is to:

- input **up to** four team names (max. 15 characters long) each with the total number match wins, draws and losses so far.
- Calculate the total aggregate point for each team based on 3 points for every win, 1 point for every draw and 0 points for every loss.
- Display on screen:
  - the highest point with the team name for today
  - a message saying whether or not the highest point today beat the current team with the highest aggregate points.
- update the file TOP_TEAM.txt if a higher aggregate point was computed today.

Task 1.1
Write program code for this task.

Evidence 1: Your program code. [8]

Task 1.2
Draw up a set of test data which tests the functioning of your program. Consider carefully all cases which could occur for both the data input and the two processing requirements.

Evidence 2: A screenshot for each test case you considered. Annotate the screenshot explaining the purpose of each test. [8]

(a) Write the league. [6]

(b) During the coding phase of the development, it is common to encounter program errors. Explain the

(i) Create a decision table that shows all possible outcomes for the above loyalty program.

2 The following is a pseudocode algorithm of a binary search written by a student to search for an item in an array `Sport`. This array stores string data and has a final subscript `MAX`.
The algorithm is poorly designed and does not work properly.

```
Enter k
A = 1
B = MAX
While not found
     C = (A+B) DIV 2
     If Sport(C) = k
          Print "Found"
     Else If Sport(C) < k
          B = C-1
     Else
          A = C+1
     End-If-Else
End-While
```

Task 2.1

(a) Identify the tables that will give a normalised solution for this problem. Draw an E-R diagram that sho

Write program code for this algorithm including all the changes you would make to:

(b) Give the table descriptions for the tables specified in part (a). [5]

- follow good programming practice
- make the algorithm work properly

Use the sample array data available from text file `SPORT+TEAM.txt` and paste this into your programming code.

Evidence 3: Your program code. [7]

Task 2.2

(a)
(b) The binary search code could be useful for many programs where a search routine is
(c) required. Re-design the program code to have a procedure `BinarySearch`. This
(d) procedure should have parameters which allow it to be used for <u>any</u> sorted array of string
(e) data.
(f) Use the data provided in the array `Team` and test the procedure with appropriate test cases
(g) to ensure it is working properly.

Evidence 4: Your amended program code. [4]

Evidence 5: A screenshot for each appropriate test case. [3]

3 An application is to be created to store a Football League table data. The total number of teams in the league is around 20. The team names are stored in the file `TEAMS.txt`.
The results of the football matches are provided in file `RESULTS.txt`. Each match data takes up one line, for example:
MadUnited 2 Chelsand 1

That is, Mad United won Chelsand, scoring two goals and conceding one goal, or Chelsand lost to Mad United, scoring one goal and conceding two goals.

The League table that needs to be created has the following information:

Team	P	W	D	L	GF	GA	GD	Points
MadUnited	3	2	1	0	5	2	3	7
Chelsand	3	2	1	0	4	2	2	7

…

Legend:
P – games played
W – games won
D – games drawn
L – games lost
GF – goals for (scored against opponents)
GA – goals against (goals conceded by team)
GD – goal difference, i.e. GD = GF – GA
Points – computed based on 3 points per win, 1 point per draw and zero points per loss

Task 3.1
Write program code for a procedure `CreateUpdateFile` which does the following:

- the program reads the **first** match results from `RESULTS.txt`
- appends to the results of each team to a text file `NEWFILE` with the following information: team name, result of match (W/D/L), goals for (GF), goals against (GA)
- for e.g. the data "MadUnited 2 Chelsand 1" will result in the following two records to be appended to `NEWFILE`,
  MadUnited,W,2,1
  Chelsand,L,1,2

Evidence 6: Your `CreateUpdateFile` program code. [8]

Task 3.2
Amend your `CreateUpdateFile` program code from Task 3.1 so that all the match results are read from `RESULTS.txt`, and the `NEWFILE` updated accordingly.

Evidence 7: Your program code for the amended procedure `CreateUpdateFile`. [6]

Task 3.3
Write program code for a function `ComputeTeamStat` which does the following:

- receives a team name as a parameter
- the function searches the file `NEWFILE` for all occurrences of that team
- calculates and outputs the team's league table information, e.g. for team 'MadUnited', it may output the following:

Team	**P**	**W**	**D**	**L**	**GF**	**GA**	**GD**	**Points**
MadUnited	3	2	1	0	5	2	3	7

Evidence 8: Your `ComputeTeamStat` program code. [8]

Evidence 9: A screenshot showing the output for the team **`Chelsand`**. [4]

Task 3.4
Write program code for a procedure `GenerateTable` which does the following:

- reads the data from files `TEAMS.txt` and `NEWFILE`
- and makes use of the function `ComputeTeamStat` from Task 3.3
- to output the complete league table information.

Evidence 10: Your `GenerateTable` program code. [8]

Evidence 11: A screenshot showing the output for the complete League table. [4]

Task 3.5
Amend your `GenerateTable` program code from Task 3.4 to output the complete League table ordered by the team with the highest points first.

Evidence 12: Your program code for the amended procedure `GenerateTable`. [2]

Evidence 13: A screenshot showing the output for the complete League table. [2]

4 Morse Code is a type of code in which letters are represented by combinations by long or short signals, e.g. the letter 'A' is represented by a short followed by a long signal, i.e. '**.-**'

Here we use the period ('.') to represent the short signal and the dash ('-') to present the long signal. The Morse code equivalent for the letters 'A' to 'Z' is provided in the file `MORSE.txt`. In this implementation, we use a space (' ') to simulate the inter-character gap and the slash ('/') to represent the inter-word gap.

Task 4.1
Write the program code for a function that will convert a given word into its Morse Code equivalent using the following specification.

```
FUNCTION ConvertWord(SingleWord:STRING):STRING
```

The function has a single parameter `SingleWord` and returns the Morse Code equivalent for that word as a string.

Evidence 14: Your `ConvertWord` program code. [8]

Evidence 15: A screenshot showing the correct Morse Code equivalent for the word "COMPUTING". [2]

Task 4.2
Write the program code that does the following:
- the user inputs a sentence of words not exceeding 10 words.
- uses the function ConvertWord in Task 4.1
- outputs the Morse Code equivalent of the sentence of words that was entered

Evidence 16: Your program code. [6]

Task 4.3
Draw up **two** suitable tests to assess that your program is working properly, explaining the reason for each test and provide screenshot evidence for your testing.

Evidence 17: Annotated screenshots for each test data run. [4]

Task 4.4
Write the program code that will do the following:
- the user enters a word string in Morse Code
- the program converts the Morse Code word to its alphabetical equivalent
- outputs the converted word

Evidence 18: Your program code. [7]

Evidence 19: A screenshot showing the correct word equivalent for the following Morse Code string, "… --- …" [1]

~~ END OF PAPER ~~