Name:		Index Number:		Class:	

DUNMAN HIGH SCHOOL
Preliminary Examination
Year 6

COMPUTING **9597**

(Higher 2)
Paper 1

28 August 2017
3 hours 15 minutes

Additional Materials: Data files

READ THESE INSTRUCTIONS FIRST

Type in the `EVIDENCE.docx` 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 marks is given in brackets [ ] at the end of each task.

Note: You may not use the built-in `sort()`, `min()`, `max()`, `sum()` and `mean()` functions.

Copy and paste required evidence of program code and screen shots into the appropriate cells in `EVIDENCE.docx`.

Data files

Q1 – `SEAGAMES2007_2015.csv`
Q3 – `LOAN.txt`
Q4 – `CORPUS.txt`

1. The variable length record file `SEAGAMES2007_2015.csv` contains the results of SEA Games from 2007 to 2015.

Task 1.1
Determine the country with the most number of medals and the corresponding medal count during this period.

Sample output:
```
Most number of medals (999): CountryX
```

Evidence 1
Program code. [6]

Evidence 2
Screenshot. [1]

Task 1.2
Determine the year(s) in which the host country also topped the medal rally (i.e. won the most number of gold medals).

Sample output:
```
CountryY (999 Gold medals): YearZ
```

Evidence 3
Program code. [7]

Evidence 4
Screenshot. [1]

2. The following shows an incomplete recursive linear search algorithm.

```
LinearSearch(data, target, first):
if # base case 1 (not found)
    return -1
if target == data[0]: # base case 2 (found)
    return first
# recursive case
return LinearSearch(data[1:], target, ?)
```

The third parameter `first` keeps track of the original index of the first item in the list `data`.

Task 2.1
Write program code to implement the above recursive search algorithm. Test your implementation with appropriate test data.

Evidence 5
Program code. [5]

Evidence 6
Screenshot. [2]

Task 2.2
Make slight modifications to make your recursive linear search more efficient. You should provide appropriate comments to explain how and why your improved recursive linear search is more efficient.

Evidence 7
Program code. [6]

Evidence 8
Screenshots. [2]

3. In line with Singapore's Smart Nation vision, Dunman Pre-School library decides to go digital to convert their physical books collection to e-books. You have been tasked to organise book and borrower information using your knowledge and skills in OOP and data structures.

The library will maintain the books information using a binary search tree (BST). Each BST node contains BookID, Title, OnLoan (the number of copies currently on loan). For copyright reason, the library will make available 3 digital copies of each title for loan. Each BST node will thus contain an array of borrower's details: BorrowerID, BorrowDate, ReturnDate. The default loan period is 10 days. Books which are overdue incur a fine of $0.15 per day.

When a book is loaned, an entry is made in an available array position. When it is Returned, the BorrowerID value will be set to -1. Items which are returned will also have the information appended to a serial fixed length record file `TRANSACTION.txt` with the following structure (Dates are stored in YYYYMMDD format):

```
<ReturnDate><BorrowerID><BookID><BorrowDate>
```

Task 3.1
Using OOP techniques, setup the BST structure and perform insertion of books information from `BOOKS.txt` You should take the necessary preprocessing step to ensure optimal search performance.

Evidence 9
Program code. [8]

Task 3.2
Write additional program code to update the BST with loan information from `LOAN.txt` which has the following structure:

```
<Action><BorrowerID><BookID><BorrowDate/ReturnDate>
```

where Action has two possible options: `B` (borrow) and `R` (return), as well as search capabilities for book and borrower.

Evidence 10
Program code. [10]

Evidence 11
Screenshot for output on query for BookIDs `978-0-9746475-0-0` and `978-1-942824-04-4`. [2]

Evidence 12
Screenshot for output on query for BorrowerIDs `T1276249B` and `T1395320H`. [2]

Task 3.3
Write program code to generate a daily report of borrowers who have overdue books and output this information to the text file `OVERDUE.txt` with the following structure:

```
<CurrentDate>
<BorrowerID><LoanDate><ReturnDate><BookID>
<FineDue>
```

The information should be ordered by BorrowerID, followed by descending order of days overdue, and lastly by BookID. Take CurrentDate to be 20170602.

Evidence 13
Program code. [8]

Evidence 14
Screenshot of `OVERDUE.txt`. [1]

Task 3.4
To make the program more robust, write validation code for `BorrowerID` which is a user's NRIC. For NRIC number in the format SABCDEFG, where A is the first digit, B is the second digit and so on.
1. sum = (2*A) + (7*B) + (6*C) + (5*D) + (4*E) + (3*F) + (2*G)
2. Add 4 to sum if your IC number starts with T or G.
3. Find the remainder when sum is divided by 11.
4. Convert the remainder into a letter using the following table.

Remainder	0	1	2	3	4	5	6	7	8	9	10
Letter (NRIC)	J	Z	I	H	G	F	E	D	C	B	A
Letter (FIN)	X	W	U	T	R	Q	P	N	M	L	K

Evidence 15
Program code. [6]

Evidence 16
Screenshot of output for NRICs `T1276249B` and `G2873148M`. [2]

Task 3.5
Write program code to determine the top 3 borrowers' IDs and top 3 most popular book titles with cumulative number of loans..

Evidence 17
Program code. [7]

Evidence 18
Screenshot. [2]

4. You have been tasked to develop a machine learning algorithm for text analysis. You have been provided with the following training data (in `CORPUS.txt`):

```
I love Computing. Computing rocks, rocks, rocks!
I have a dog and a cat.
Best of Computing? Projects, projects, projects!
My cat keeps chasing my dog. Cats!
```

Task 4.1
Create and output a frequency dictionary of case-insensitive key terms for the corpus. For the above data, after removing stop words (eg I, have, etc.) the key terms will be

```
best, cat, cats, chasing, computing, dog, love, projects, rocks
```

Evidence 19
Program code. [5]

Evidence 20
Screenshot of dictionary contents. [1]

Task 4.2
Create a 2D array to store the frequency of sorted key terms in each sentence. For example, for the first sentence, the array contents will be:

```
[0, 0, 0, 0, 2, 0, 1, 0, 3]
```

Evidence 21
Program code. [4]

Evidence 22
Screenshot. [2]

Task 4.3
Update your 2D array to give a normalised frequency of each key term i.e. divide the number of occurrences of each key term in a sentence by the total number of key terms in the sentence. This gives a more accurate representation of the importance of the key terms relative to the length of the sentence. For the first sentence, the updated weighted array contents will be:

```
[0, 0, 0, 0, 0.333, 0, 0.167, 0, 0.5]
```

Output your normalised array.

Evidence 23
Program code. [5]

Evidence 24
Screenshot. [2]

Task 4.4
Use your normalised array and an appropriate algorithm, divide the key terms into 2 groups. Output the contents of the 2 groups.

Evidence 25
Program code. [9]

Evidence 26
Screenshot. [2]

***** THE END *****