Name:		Index Number:		Class:	

DUNMAN HIGH SCHOOL
Preliminary Examination
Year 6

COMPUTING **9597**

(Higher 2)
Paper 1

31 August 2015
3 hours 15 minutes

Additional Materials: Data files and `EVIDENCE.docx`

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.

Copy and paste required evidence of program code and screen shots into `EVIDENCE.docx`.

At the end of the examination, print out and submit your `EVIDENCE.docx`.

Data files

Q1 – `ELECTION.txt`
Q2 – `PARTICIPANTS.txt`
Q3 – `Nil`
Q4 – `Nil`

1. The coming General Election will see 28481 voters in the MacPherson Single Member Constituency (SMC) voting for their Member of Parliament (MP). The data file `ELECTION.txt` contains the simulated voting results, with A, B, C and V representing votes for the political parties National Solidarity Party, People Action Party and Workers' Party, as well as void votes respectively.

Task 1.1
Determine the winner and the voting statistics (correct to 1 decimal place) for this SMC. Your output should look like the following:

Sample output:

```
Results for the Electoral Division of MacPherson
PAP  58.7%
WP   37.5%
NSP   2.6%
Void  1.2%
Winner is PAP
```

Evidence 1
Program code. [7]

Evidence 2
Screenshot. [1]

Task 1.2
In the event of a tie, the parties tied will enter into a revoting exercise. For example, if there is a tie between PAP and WP, return the message "`Revote between PAP and WP`".
Write program code to generate a text file `TIE.txt` with approximately 3% void votes, 7% votes for NSP and a tie between PAP and WP. Test your program with your generated `TIE.txt`.

Evidence 3
Program code. [6]

Evidence 4
Screenshot. [1]

2. You are provided with the registration dates and email addresses of participants who registered for an event in the text file `PARTICIPANTS.txt`. Unfortunately, some participants registered more than once.

Task 2.1
Write program code to remove duplicate entries and generate the final participant list in alphabetical email order. If a participant registered more than once, take the last registered record as the final entry. Return also a count of duplicate entries removed.

Sample output:
```
ali@yahoo.com,03-05-2015
dave@savetheworld.org,05-05-2015
james007@bond.uk,07-05-2015
limahseng@gmail.com,01-05-2015
mary_ong@m1.com.sg,05-05-2015
zorro@outlook.com,30-04-2015

3 duplicate entries removed.
```

Evidence 5
Program code. [4]

Evidence 6
Screenshot. [1]

Task 2.2
Some email addresses may be entered wrongly. Using first principles, write a Boolean function `Valid_Email(email)` to determine if an email address is valid. Design test cases to thoroughly test the working of your function code.

Evidence 7
Program code. [6]

Evidence 8
Test data with purpose and corresponding screenshots. [4]

3. International Standard Book Number (ISBN) is a unique number assigned to each edition of a book. It has two formats: ISBN-10 and ISBN-13. The former is used before 1 Jan 2007 and the latter after that. Examples of valid ISBNs are 020103803X and 978-1-284-05591-7. The last digit is the check digit and is computed as follows:

ISBN check digit (10 digits) - mod 11 algorithm

- Each digit starting from left to right is assigned a weight from 1 to 9. Each digit is multiplied by its position weight. The sum of products modulo 11 gives a remainder between 0 and 10. If the remainder is 10, the check digit is the roman numeral X, else the check digit is the remainder.
- **Example ISBN-10: 0-07-063546-3**
- (0x1) + (0x2) + (7x3) + (0x4) + (6x5) + (3x6) + (5x7) + (4x8) + (6x9) = 190
- 190 mod 11 = 3
- Hence 3 is the check digit.

ISBN check digit (13 digits) - mod 10 algorithm

- Each digit starting from the left to right is multiplied by 1 or 3 alternatively. The sum of the products modulo 10 gives a remainder between 0 to 9. If the remainder is non-zero, subtract this from 10 to get the check digit, else the check digit is 0.
- **Example ISBN-13: 978-0-07-063546-3**
  1x9 + 3x7 + 1x8 + 3x0 + 1x0 + 3x7 + 1x0 + 3x6 + 1x3 + 3x5 + 1x4 + 3x6 = 117
- 117 mod 10 = 7
- 10 - 7 = 3
- Hence 3 is the check digit.

Task 3.1
Write a function `ISBN_Check_Digit(isbn)` which generates the check digit for a given ISBN-10 or ISBN-13 number.

Evidence 9
Program code. [5]

Evidence 10
Screenshots (1 for ISBN-10 and 1 for ISBN-13) [2]

Task 3.2
Write a function `Valid_ISBN(isbn)` which calls your `ISBN_Check_Digit(isbn)` function in Task 3.1 to determine if a given ISBN-10 or ISBN-13 number is valid.

Evidence 11
Program code. [4]

Evidence 12
Screenshots for appropriately generated ISBN-10 and ISBN-13 numbers. [2]

An ISBN-10 number can be converted to its ISBN-13 equivalent by prefixing '978' to the ISBN-10 number and then calculating the check digit using the ISBN-13 algorithm.

Task 3.3
Write a function `ISBN10_To_ISBN13(isbn)` to convert an ISBN-10 number to its ISBN-13 equivalent.

Evidence 13
Program code. [3]

Evidence 14
Screenshot of output for ISBN-10 number 1904467520. [1]

Task 3.4
Write a function `ISBN13_To_ISBN10(isbn)` to convert an ISBN-13 number to its ISBN-10 equivalent.

Evidence 15
Program code. [4]

Evidence 16
Screenshot of output for ISBN-13 number 9780748740468. [1]

A small library wishes to store its book collection details using a random file organisation. It currently holds 200 books but wishes to expand its collection by about 10% every year. To resolve collisions, it decides to use double hashing which minimises clustering. For a start, assume that this file organisation will be maintained for 3 years.

Task 3.5
Devise and implement a suitable double hashing strategy `Hash_Key(isbn)`. Annotate your strategy using program comments.

Evidence 17
Program code with comments. [5]

Task 3.6
Write program code to generate an appropriate number of ISBNs to the random text file `LIBRARY.txt` which will cater to the library's collection growth over 3 years. Include `Insert_Book(isbn)` and `Lookup_Book(isbn)` functions to insert a book and lookup a book respectively.

Evidence 18
Program code. [8]

Evidence 19
Screenshots showing the insertion and lookup of one non-collided record and one collided record. [4]

Evidence 20
Separate printout of `LIBRARY.txt` after insertion of all generated records. [2]

4. You are tasked to maintain a binary search tree for optimal search efficiency. Each node of the binary search tree contains a unique identifier (productid) and a reference to a linked list which holds individual order details for that product. For simplicity, orderid will suffice.

Task 4.1
Using object-oriented programming, implement the above binary search tree with its associated class methods (`insert()`, `search()` and `inorder()`).

Evidence 21
Program code for binary search tree with 8 products inserted, 3 of which has at least 2 orders. [8]

Evidence 22
Screenshot of output of inorder traversal (product and order information). [2]

Task 4.2
Write a support class method `most_popular()` to determine the best selling product(s).

Evidence 23
Program code. [4]

Evidence 24
Screenshot. [1]

Task 4.3
A product is no longer available. Write a class method `delete()` to remove this node from the binary search tree, accounting for all possible cases.

Evidence 25
Program code. [5]

Evidence 26
Screenshots. [3]

Task 4.4
After a series of insertion and deletion, the binary search tree may become unbalanced and this requires reorganisation to maintain optimal search efficiency. Write functions `is_balanced()` to test if a binary search tree is balanced, and `reorg()` to reorganise an unbalanced binary search tree. Include appropriate annotation.

Evidence 27
Program code. [7]