MINISTRY OF EDUCATION, SINGAPORE
in collaboration with
UNIVERSITY OF CAMBRIDGE LOCAL EXAMINATIONS SYNDICATE
General Certificate of Education Advanced Level
Higher 2

COMPUTING

9597/01

Paper 1

October/November 2016

3 hours 15 minutes

Additional Materials: Pre-printed A4 paper
Removable storage device
Electronic version of KEYS1.TXT data file
Electronic version of KEYS2.TXT data file
Electronic version of EVIDENCE.DOC document

0962790389

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 screenshots into the EVIDENCE.DOC document.

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

This document consists of **8** printed pages.

Singapore Examinations and Assessment Board

CAMBRIDGE International Examinations

1 High-level programming languages usually have libraries of commonly used routines. These include random number generators.

Task 1.1

Write program code to generate 1000 random integers in the range 1 to 20.
The program will:

- Maintain a count of how many times each number is produced
- Print out a frequency table.

Example output:

Integer	Frequency
1:	54
2:	48
3:	52
4:	43
5:	48
6:	51
7:	41
8:	48
9:	53
10:	51
11:	45
12:	54
13:	44
14:	40
15:	54
16:	59
17:	47
18:	49
19:	66
20:	53

Evidence 1
Your program code.
Screenshot of the program output. [7]

Random numbers generated by computers are usually referred to as pseudo-random numbers because they are generated by executing program code.

One criterion of a good pseudo-random number generator is that every number in the range has an equal chance of being generated. This means if 200 numbers are generated in the range 1 to 10, the expected frequency value of every number in this range is 20.

The program code is to be amended to check how well the given pseudo-random number generator meets this requirement.

Task 1.2

Amend your program code to:

- Calculate the expected frequency
- Output this expected frequency
- Output the difference between the actual and the expected frequency for each number in the range as a third column of the frequency table.

Evidence 2

Your program code.
Screenshot of the program output. [5]

2 Customers are identified by ID numbers. These ID numbers are to be stored in a hash table. The hashing function to be used is

```
Address ← IDnumber MOD Max
```

The hash table is implemented as a one-dimensional array with elements indexed 0 to (`Max-1`).

Task 2.1

Write program code to:

- Read ID numbers from a text file and store them in a hash table. For the purpose of testing the program, `Max` is to be set to the value 20.
  Assume different IDs will hash to different addresses (no collisions).
- Print out the contents of the hash table in the order in which the elements are stored in the array.

Use `KEYS1.TXT` to test your program code.

Evidence 3
Your program code.
Screenshot of the program output. [7]

Task 2.2

Amend your program code so that collisions can be managed using open hashing. This means a collision is resolved by searching sequentially from the hashed address for an empty location and storing the ID at this empty location.

Use `KEYS2.TXT` to test your program code.

Evidence 4
Your program code.
Screenshot of the program output. [4]

Task 2.3

Add code to your Task 2.2 program.
The program is to:

- Take as input an ID number
- Search the hash table and output the address (index number) of the hash table where the ID was found.

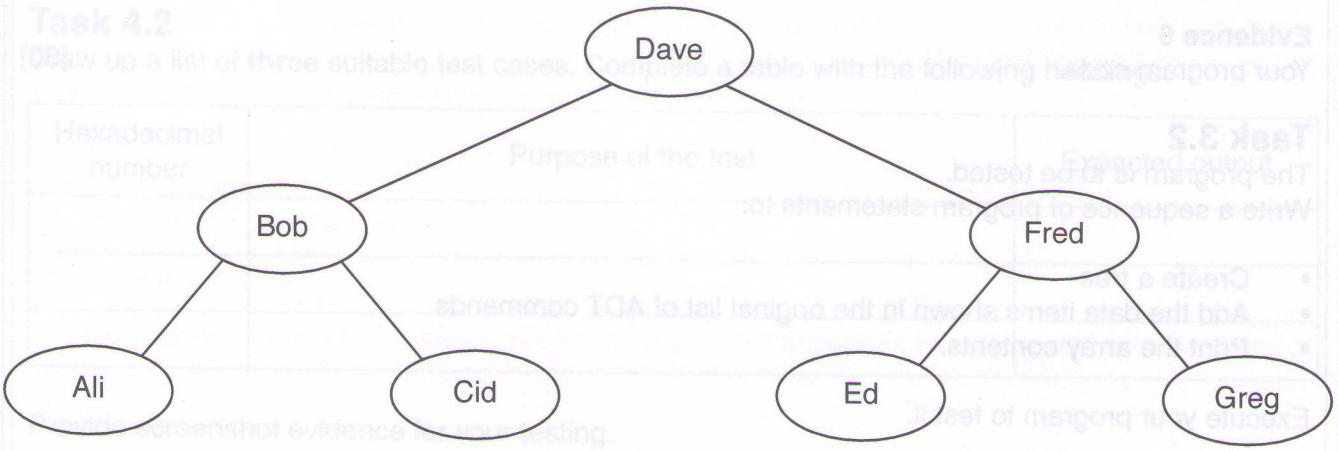
Use `KEYS2.TXT` to test your program code.
Run the program three times. Use the following inputs: 37, 77 and 97.

Evidence 5
Your program code.
Screenshot of the program output. [7]

3 A binary tree Abstract Data Type (ADT) has commands to create a new tree, add unique data items to the tree and print the tree.
The sequence of commands:

```
CreateNewTree
AddToTree("Dave")
AddToTree("Fred")
AddToTree("Ed")
AddToTree("Greg")
AddToTree("Bob")
AddToTree("Cid")
AddToTree("Ali")
```

would create the following binary tree:



The program to implement this ADT will use the classes `Tree` and `Node` designed as follows:

Tree
`tree : ARRAY OF Node` `root : INTEGER`

`constructor()`
`add(newItem)`
`print()`

Node

`data : STRING`
`leftPtr : INTEGER`
`rightPtr : INTEGER`

`constructor()`
`setData(s : STRING)`
`setLeftPtr(x : INTEGER)`
`setRightPtr(y : INTEGER)`
`getData() : STRING`
`getLeftPtr() : INTEGER`
`getRightPtr() : INTEGER`

[Turn over

The program code must:

- Create a new tree, which has:
  - no nodes
  - the root set to –1
- Use the root as a pointer to the first node in the tree
- Add a new node to the tree in the appropriate position
- Use the `print()` method to output, for each node, in array order:
  - the data item
  - the left pointer
  - the right pointer.

Task 3.1

Write program code to define the classes `Tree` and `Node`.

Evidence 6
Your program code. [30]

Task 3.2

The program is to be tested.
Write a sequence of program statements to:

- Create a tree
- Add the data items shown in the original list of ADT commands
- Print the array contents.

Execute your program to test it.

Evidence 7
Your program code.
Screenshot of test run. [3]

Task 3.3

A method `inOrderTraversal()` is to be added, which outputs the data stored in the tree in alphabetical order.
Write program code to:

- Implement this method
- Test the program code with the data from Task 3.2.

Evidence 8
Your program code.
Screenshot of test run. [7]

4 Numbers in Computing are often represented in hexadecimal form.
A program is required to convert a hexadecimal number into a denary number and vice versa.

Task 4.1

Write program code with the following specification:

- Input a hexadecimal number as a string
- Validate the input
- Calculate the denary value of each hexadecimal digit (write this code as a function)
- Calculate the denary value of the hexadecimal number input
- Output the denary value.

Evidence 9
Your program code. [10]

Task 4.2

Draw up a list of **three** suitable test cases. Complete a table with the following headings:

Hexadecimal number	Purpose of the test	Expected output

Provide screenshot evidence for your testing.

Evidence 10
The completed table.
Screenshots for each test data run. [5]

Task 4.3

Write additional code to convert a denary number into a hexadecimal number.

Evidence 11
Your program code. [10]

Task 4.4

Draw up a list of **three** suitable test cases. Complete a table with the following headings:

Denary number	Purpose of the test	Expected output

Provide screenshot evidence for your testing.

Evidence 12
The completed table.
Screenshots for each test data run. [5]