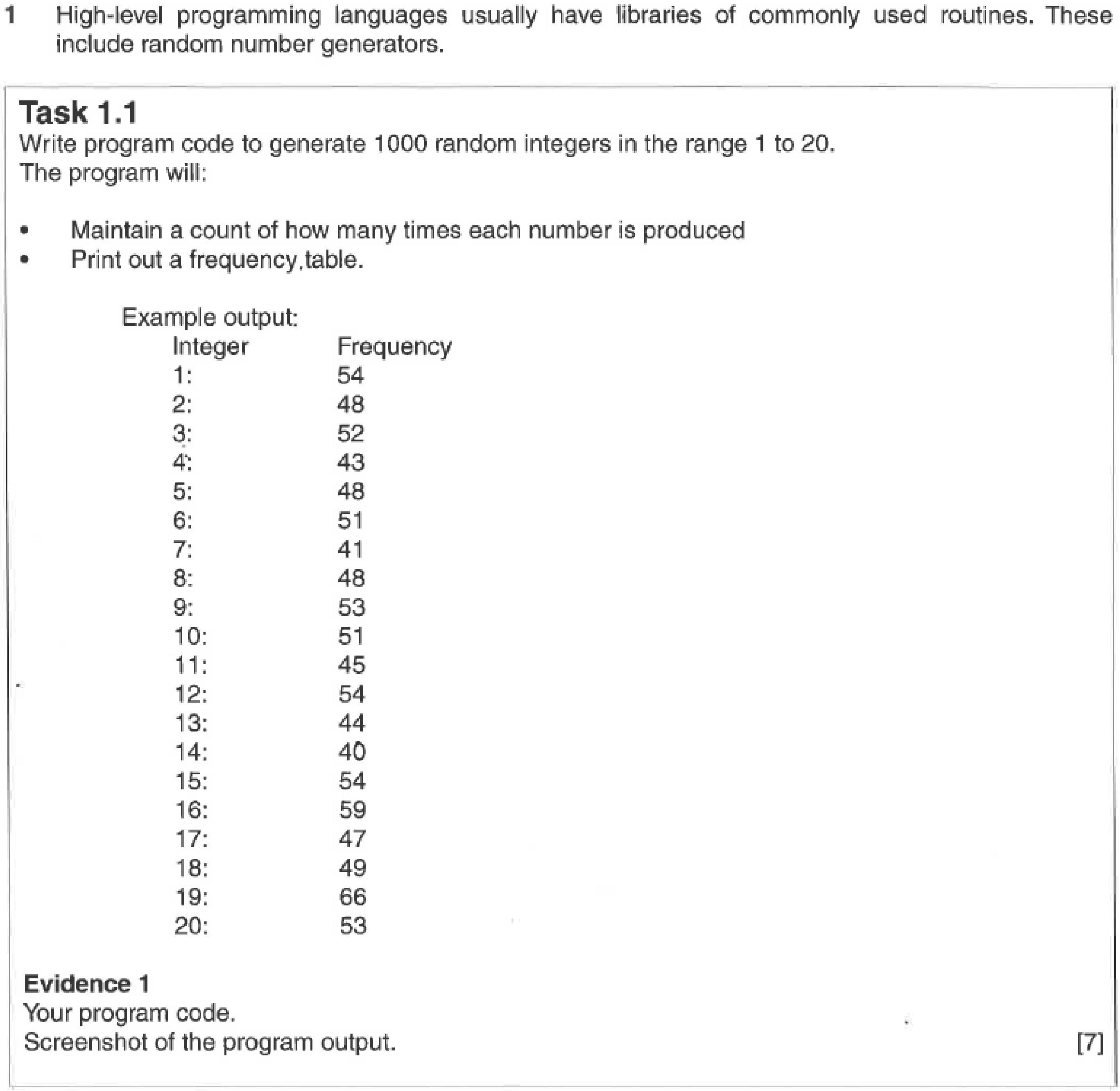
# JURONG PIONEER JUNIOR COLLEGE

## 2022 JC 2 Practical Revision Package 1

**COMPUTING** **9569/02**
**Higher 2**

**Paper 2 (Lab-based)** **3 hours**



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

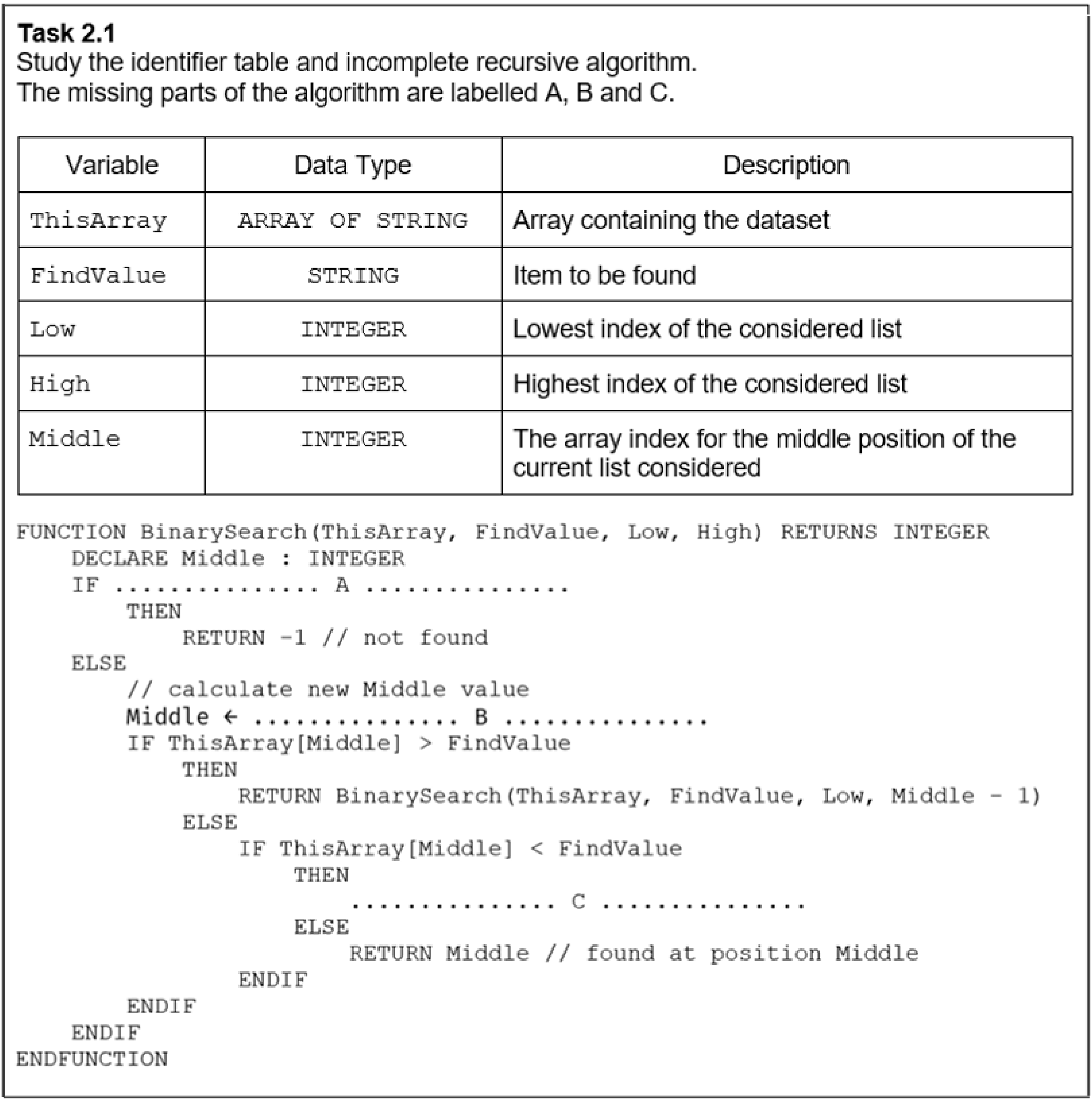
Amend your program code to:

- 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.** A binary search (binary chop) is a technique to search for a value in an ordered dataset.



**Task 2.1**
Study the identifier table and incomplete recursive algorithm.
The missing parts of the algorithm are labelled A, B and C.

| Variable | Data Type | Description |
|---|---|---|
| `ThisArray` | `ARRAY OF STRING` | Array containing the dataset |
| `FindValue` | `STRING` | Item to be found |
| `Low` | `INTEGER` | Lowest index of the considered list |
| `High` | `INTEGER` | Highest index of the considered list |
| `Middle` | `INTEGER` | The array index for the middle position of the current list considered |

```
FUNCTION BinarySearch(ThisArray, FindValue, Low, High) RETURNS INTEGER
    DECLARE Middle : INTEGER
    IF .............. A ..............
        THEN
            RETURN -1 // not found
    ELSE
        // calculate new Middle value
        Middle ← .............. B ..............
        IF ThisArray[Middle] > FindValue
            THEN
                RETURN BinarySearch(ThisArray, FindValue, Low, Middle - 1)
            ELSE
                IF ThisArray[Middle] < FindValue
                    THEN
                        .............. C ..............
                    ELSE
                        RETURN Middle // found at position Middle
                ENDIF
        ENDIF
    ENDIF
ENDFUNCTION
```

**Evidence 3**
What are the three missing lines in this pseudocode? [3]

**Task 2.2**
Write a program to implement binary search.
The program will

- Call procedure InitialiseAnimals
- Input an animal name
- Use the function BinarySearch
- Report whether or now this animal name was found. If found, also output the index position.

The array in the program has identifier MyAnimal.
Use the dataset given in the file ANIMALS.TXT. You should paste the contents of this file into your program. The statements will form the basis of the code for the procedure InitialiseAnimals.

**Evidence 4**
Program code for Task 2.2 [7]

**Evidence 5**
Screenshot to confirm that an animal which is present in the list was found with its index position displayed. [1]

**Task 2.3**
Amend the program as follows:
The program must also output the number of function calls carried out.

**Evidence 6**
The amended program code. [4]

**Evidence 7**
Screenshots showing the amended output for runs of the program where:

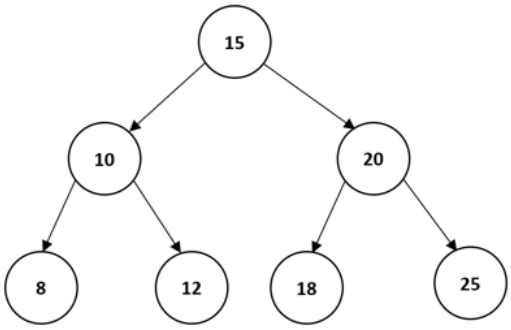
- the animal is found
- the animal is not found. [2]

**3** Create a binary tree Abstract Data Type (ADT) with commands to create a new tree, insert data items to the tree and print the tree.

The sequence of commands

Create a new tree
Add to tree (15)
Add to tree (10)
Add to tree (20)
Add to tree (8)
Add to tree (12)
Add to tree (18)
Add to tree (25)

would create the following binary tree:



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

| Tree |
|---|
| `root : Node` |
| `constructor()`<br>`add(newItem)`<br>`printTreeInOrder()` |

| Node |
|---|
| `key   : INTEGER`<br>`left  : Node`<br>`right : Node` |
| `constructor()`<br>`insert(key : INTEGER)` |

**Task 3.1**
Write program code to define the classes `Tree` and `Node.`

**Evidence 8**: Your program code. [16]

**Task 3.2**

- Write program code for a procedure `CreateTreefromArray` that accepts an array of unsorted unique integers passed in via a parameter.
- The procedure will read each integer in the array and construct a binary tree using your classes `Tree` and `Node`.
- Call `printTreeInOrder` to display the output (numbers shown will always be sorted).
- Test your program by copying the input data found in `BST.txt` into your code.

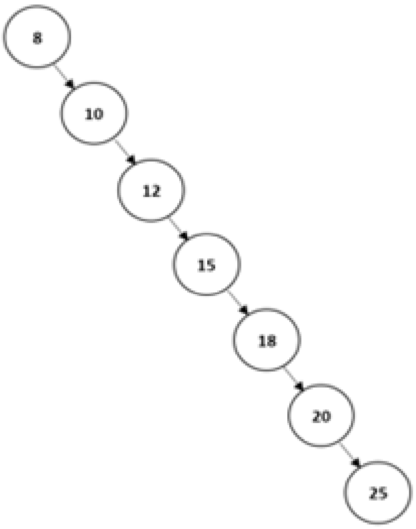
**Evidence 9:** Your `CreateTreefromArray` program code. [6]

**Evidence 10:** A screenshot of the output. [2]

**Task 3.3**
A binary tree created from keys that are in ascending order will result in an unbalanced binary tree.

For instance, the sequence of commands

Create a new tree
Add to tree (8)
Add to tree (10)
Add to tree (12)
Add to tree (15)
Add to tree (18)
Add to tree (20)
Add to tree (25)



Amend procedure `CreateTreefromArray` so that the created tree from any input array of integers will be balanced where the number of items on the left and right subtree will roughly be divided equally (Hint: input array must first be sorted).

**Evidence 11:** Your amended program code. [7]

**Task 3.4**
Create a function `FindKthSmallest` that returns the k^th^ smallest element in your binary tree. If k = 5 the k^th^ smallest element will be 18. Your function should not need to use extra space (e.g. creating a new array) to solve the problem other than using a temp variable(s).

**Evidence 12**: Your program code for `FindKthSmallest`. [7]

**Evidence 13**: Produce a screenshot showing the retrieval of the 5th smallest element from the tree created earlier. [2]

4. The examinations department of a school needs to keep long-term records of the overall examination achievements of its students.

Students at the school have two main choices. Firstly, they can take a variety of subjects and achieve an Academic Diploma. A diploma gives them the opportunity to go to university. Secondly they can achieve a Skills Certificate where they focus on one particular area (such as IT). This gives them the necessary skills to start a career in their chosen area.

The examinations department decides to store the following data:

- `StudID` is used to uniquely identify a particular student and is six digits. The first four digits represent the year that the student started at the school and the last two digits are used to make the `StudID` unique e.g. `201804`.

- `Name` is the name of the student and is at most 30 characters.

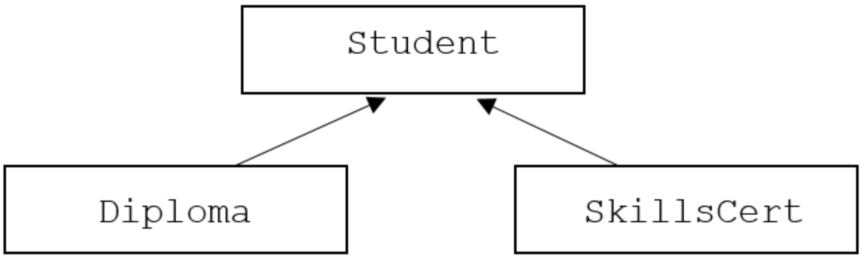
- `StudType` is the type of student and can have the values '`D`' or '`S`'.

- `SkillArea` is text and gives the area that the student acquired skills in. It can have one of three values: '`IT`', '`Business`', or '`Accountancy`'.

- `NoOfSub` is the number of subjects studied by those taking the Diploma.

- `Result` is a single character and is used to indicate the overall grade awarded. For those students who took the Skills Certificate the grades could be Distinction (`D`), Merit (`M`), Pass (`P`) or Fail (`F`). For those who took the Diploma the grade could be one of the letters `A` to `F`. Grade `A` to `E` are passes. Grade `F` is a fail.

The program design for a solution to this problem is to be implemented with object-oriented programming with the following three classes:



**Task 4.1**
Write program code to define the classes `Student`, `Diploma` and `SkillsCert`.

**Evidence 14**
Program code for the three classes in Task 4.1. [10]

Assume that a file, `STUDENT.txt`, which contains details of each student, has been created for you. The format of each student record is as follows:

```
<StudID>|<Name>|<StudType>|<SkillArea>|<NoOfSub>|<Result>
```

- `SkillArea` would have the value '`Diploma`' if the student is taking a Diploma.
- `NoOfSub` would have the value `0` for those taking the Skills Certificate.
- `Result` is left blank initially.

**Task 4.2**
Write a module, `ENTER_RESULT`, which, when called, will ask the user for a particular `StudID` whose result is to be entered. Using the student ID that has been input, the corresponding student record will be located in `STUDENT.txt`. The student data will be displayed to the user. The user will be allowed to enter the result for the student. The amended record will be stored back in `STUDENT.txt`.

The student ID and result that have been input should be validated.
If the `StudID` does not exist, the user will be given an appropriate message.

**You are expected to make use of the classes you designed in Task 4.1.**

Run the program **three** times. Use the following data input, and produce a screenshot for each.

| StudID | Result |
|---|---|
| 201701 | A |
| 201801 | B |
| 201901 | M |

**Evidence 15**
Program code for Task 3.2 [8]

**Evidence 16**
**Three** screenshots showing the test runs and final contents of `STUDENT.txt` to show evidence that successful updates have been carried out. [3]

**Task 4.3**
Implement code as specified below.

A report should be generated and displayed which will list the students whose result has still not been entered into the `STUDENT.txt` file. The report will list, for each different starting year:

- `StudID`
- `Name`
- `StudType`
- `SkillArea` or `NoOfSub` depending upon the value of `StudType`

In addition the number of each student type for each year will also be output.

```
     Year: 2017
-----------------------------------------------------
     201715    FLoo           D     6
     201708    BLang          D     5
     201710    LArms          S     IT
     Diplomas:                2
     Skills:                  1

     Year: 2018
-----------------------------------------------------
     201813    EJean          D     7
     201817    ABright        D     7
     Diplomas:                2
     Skills:                  0

     Year: 2019
-----------------------------------------------------
     201905    Alfie          S     Business
     201903    GKoh           D     8
     Diplomas:                1
     Skills:                  1
```

**Evidence 17**
Program code for Task 4.3. [8]

**Evidence 18**
Screenshot of the output produced. [2]