Name: ……………………………………………….. ( ) Class: ……… Date: ………………….

|  |  |  |
| --- | --- | --- |
| **3.1** | **Problem Analysis** | **Modularity and Generalisation** |

**Understanding Algorithms**

1. What is the output of the following algorithm if the input word is “COMPUTING”?

**Step 1:**  Swap the first and last letters of the word.

**Step 2:**  Reverse the letters of the word.

**Step 3:**  If the second letter of the word is “N”, remove it and join the remaining letters back into a single word.

**Step 4:** If the fifth letter of the word is “I”, add “C” to the end of the word.

**Step 5:**  Insert “H” between the second and third letters of the word.

**Step 6:**  Remove the fifth to seventh letters of the word and join the remaining letters back into a single word.

**Step 7:**  Reverse the letters of the word and output the result.

………………………………………………………………………………………………………………………………………………………

**Decomposition**

Two common approaches to decomposing a problem are:

1. Incremental approach: Identify quantitative features of the input or output that cause the problem to be too large to handle. Sometimes, the solution to a small version of the problem with one or more of these features reduced can be found and gradually extended to larger versions of the problem. Each gradual extension of the solution is a separate sub-problem.
2. Modular approach: Solve simple examples of the problem manually and identify tasks that are of different nature. Usually, these tasks can be separated from each other to become distinct (and sometimes unrelated) sub-problems. This usually results in sub-problems that are different from each other.
3. You wish to sort the students of your class in ascending order of height. Why is this problem difficult to solve in one step?

**A** There are many classes in your school

**B** There are many students in your class

**C** There are many students in your school

**D** The tallest student in your class is very tall ( )

1. You wish to find out whether the tallest student in your class has signed up for the basketball event in your school’s Games Day. (Assume that that no two students in your class have the same height.) Using a modular approach, which set of sub-problems below is the most useful decomposition of this problem to obtain the required output?

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **Sub-Problem** | **Input** | **Output** |
|  | #1 | * *Heights*: table of NRICs for students in your class and their corresponding heights | * *Tallest height:* tallest height value in *heights* |
|  | #2 | * *Height*: height of a student * *Basketball list*: list of NRICs for students in your school who signed up for basketball and their corresponding heights | * Whether there is a student in *basketball list* with given *height* or higher (must be either “Yes” or “No”) |

|  |  |  |  |
| --- | --- | --- | --- |
| **B** | **Sub-Problem** | **Input** | **Output** |
|  | #1 | * *Heights*: table of NRICs for students in your class and their corresponding heights | * *Tallest student:* NRIC of tallest student in *heights* |
|  | #2 | * *Student*: a student’s NRIC * *Basketball list*: list of NRICs for students in your school who signed up for basketball | * Whether *student* is in *basketball list* (must be either “Yes” or “No”) |

|  |  |  |  |
| --- | --- | --- | --- |
| **C** | **Sub-Problem** | **Input** | **Output** |
|  | #1 | * *Class list*: list of NRICs for students in your class * *Basketball list*: list of NRICs for students in your school who signed up for basketball | * *Class basketball list:* list of NRICs for students in *class list* who are also in *basketball list* |
|  | #2 | * *Class basketball list*: list of NRICs for students in *class list* who are also in *basketball list* | * NRIC of any student in *class basketball list* |

|  |  |  |  |
| --- | --- | --- | --- |
| **D** | **Sub-Problem** | **Input** | **Output** |
|  | #1 | * *Heights*: table of NRICs for students in your class and their corresponding heights * *Height*: height of a student | * *Tall student:* NRIC of any student in *heights* with given *height* or higher |
|  | #2 | * *Student*: a student’s NRIC * *Basketball list*: list of NRICs for students in your school who signed up for basketball | * Whether *student* is in *basketball list* (must be either “Yes” or “No”) |

( )

1. Assuming that no two gifts in a gift store have the same price, the problem of finding the most expensive gift being sold can be defined as:

|  |  |
| --- | --- |
| **Input** | **Output** |
| * *Price list:* list of gifts and their corresponding prices; list cannot be empty and no two gifts can have the same price | * Most expensive gift in *price list* |

Using an incremental approach, a possible way to define the first two sub-problems is:

|  |  |  |
| --- | --- | --- |
| **Sub-Problem** | **Input** | **Output** |
| #1 | * *First gift*: first gift in *price list* and its corresponding price | * *Result #1*: Most expensive gift from first item in price list |
| #2 | * *Result #1*: Most expensive gift from first item in price list * *Second gift*: second gift in *price list* and its corresponding price | * *Result #2*: Most expensive gift from first two items in price list |
| #3 |  |  |
|  | *…and so on…* | *…and so on…* |

Suggest a possible way to define sub-problem #3 in the table above.

**Generalisation**

1. The problem of obtaining the height of the tallest student can be defined as follows:

|  |  |
| --- | --- |
| **Input** | **Output** |
| * *Heights list*: list of students‘ NRICs and heights | * Tallest height value in *heights list* |

Which of the following problems is likely to have a similar solution? ( )

|  |  |  |
| --- | --- | --- |
|  | **Input** | **Output** |
| **A** | * *Word list*: list of words containing at least one word; all words must be made of English letters | * List of words in *word list* that appear more than once |
| **B** | * *Word list*: list of words containing at least one word; all words must be made of English letters | * Word that appears most often in *word list* |
| **C** | * *Word list*: list of words containing at least one word; all words must be made of English letters | * Whether the first word in *word list* is longer than words in the remainder of *word list* (must be either “Yes” or “No”) |
| **D** | * *Word list*: list of words containing at least one word; all words must be made of English letters | * Length of longest word in *word list* |

1. The problem of calculating the remainder when a number is divided by 3 can be defined as follows:

|  |  |
| --- | --- |
| **Input** | **Output** |
| * *Number:* a positive whole number | * Remainder when *number* is divided by 3 |

A possible solution to this problem is as follows:

**Step 1:** If *number* is less than 3, proceed to Step 3. Otherwise, proceed to Step 2.

**Step 2:** Subtract 3 from *number* and then proceed to Step 1.

**Step 3:** Output *number* as the final answer.

A similar problem is as follows:

|  |  |
| --- | --- |
| **Input** | **Output** |
| * *Number:* a positive whole number * *Divisor*: a positive whole number | * Remainder when *number* is divided by *divisor* |

Generalise the first solution so it solves the second problem.

………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………