DUNMAN HIGH SCHOOL
Preliminary Examination
Year 6

COMPUTING

9569 / 01

Paper 1 Written

22 Sep 2021

3 hours

READ THESE INSTRUCTIONS FIRST

An answer booklet will be provided with this question paper. You should follow all the instructions on the front cover of the answer booklet. If you need additional answer paper ask the invigilator for a continuation booklet.

Answer **all** questions.
Approved calculators are allowed.

The number of marks is given in brackets [ ] at the end of each question or part question.
The total number of marks for this paper is 100.

1 A cosmic ray striking computer memory at just the right time can flip a bit, turning a 0 into a 1 or vice versa.

Bit flips had caused plane accidents, software glitches, and at times, the Blue Screen of Death (BSoD) on personal computers.

(a) The hexadecimal number `D4` had been changed to `9A`. Suggest the minimum number of times a bit flip could have affected it and explain why. [2]

(b) Explain why Unicode characters, as compared to ASCII characters, are more likely to be altered after a cosmic ray strike. [2]

(c) Define what a backup and archive is, and explain which should be prioritised if a company is warned of a global cosmic ray strike occurring in a week. [4]

(d) Explain how TCP/IP layers help to fulfil the purpose of the TCP/IP model. [2]

(e) Suggest and explain a consequence on network communication when a data packet is affected by a bit flip occurring at the Internet Layer. [2]

(f) Explain why the consequence of a bit flip occurring at the Internet Layer and the Transport Layer would be the same as the consequence as a bit flip occurring at the Internet Layer only.

[1]

2 An array stores 16 powers of 2 integers in ascending order:

1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768

(a) If the array is searched by means of a binary search, state which elements would be accessed, and in what order,

i) when searching for the number 4096 (which is present), and [1]

ii) when searching for the number 3 (which is not present). [1]

(b) Draw a program flowchart of an iterative binary search algorithm. [6]

(c) Explain why binary search could be more efficient than linear search. [1]

(d) State the time complexities of hash table search and binary search and explain which is more efficient with reference to their time complexities in the above scenario.

[4]

3 A mobile network provider's management of customer's overdue bills include an automated emailing and SMS system.

- If a mobile bill is overdue, a daily system generated reminder is emailed to the user indicating the overdue details.
- If the user has 1 mobile bill that is more than 6 days overdue, in place of the daily reminder email, a warning email and SMS will be sent to the user.
- If a user has more than 4 bills which are more than 14 days overdue, the user will incur a penalty fee for every additional week starting from the 14th day of being overdue.
- If a user has incurred more than 3 penalty fees, the user's mobile service will be terminated.
- Penalty fees could be incurred as a result of other actions such as accessing illegal websites.
- Each bill issued to a customer represents the past month's usage.

(a) Create a decision table showing all possible outcomes and results for bill(s) that are overdue. [4]

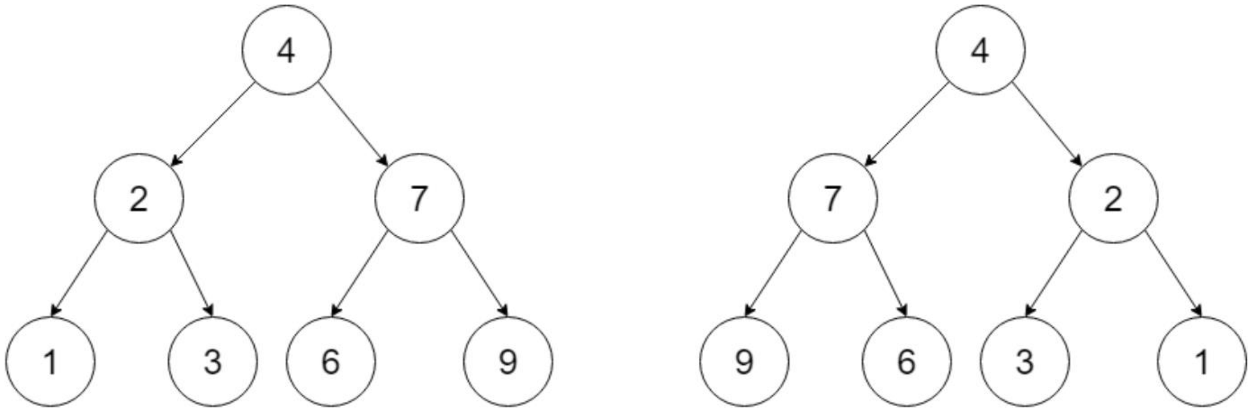
(b) Simplify the decision table by removing redundancies. [3]

(c) User interface is designed for a program for customers to check their bills and overdue status. State a usability principle and describe how the user interface of the program should be designed to demonstrate this principle. [2]

4 **(a)** Explain how a linked list data structure could be more suitable than an array data structure to implement a binary tree. [2]

(b) Suggest and justify one circumstance where an array structure is more appropriate than a linked structure. [2]

The diagram shows a binary tree before and after an inversion.



Pre-inverted tree

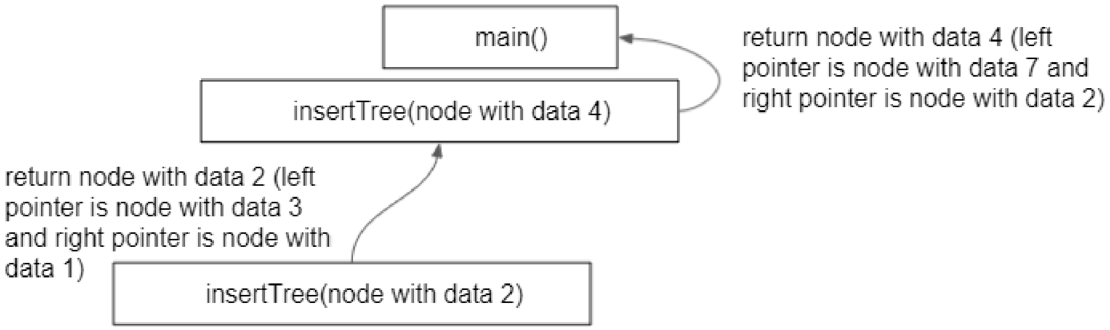
Post-inverted tree

Each node has these attributes:

- `right` which is the pointer to right subtree
- `data` which is the value contained in each respective node displayed above
- `left` which is the pointer to the left subtree

(c) Write pseudocode for procedure `insert` which will add a node to the post-inverted tree (which may be empty) in such a way that if the new value of the node is **less** than the value at the current node, the new node will be added to the **right** subtree, or else it will be added to the left subtree. `insert` takes in values `node_data` and `root_node` which is the value to be added to the tree and the instance of the root node of the tree respectively. [6]

(d) A function `invertTree` takes in the root node of the above pre-inverted tree, uses recursion to invert it into the post-inverted tree, and returns the root node of the post-inverted tree. Write pseudocode for the function and visualise it in a trace diagram. An incomplete trace diagram is provided for you to begin with. Copy and complete it.



return node with data 4 (left pointer is node with data 7 and right pointer is node with data 2)

return node with data 2 (left pointer is node with data 3 and right pointer is node with data 1)

(Note: Should be invertTree, not insertTree)

[6]

5 Consider the following data which shows a single Civics Group record used in COVID19 vaccination tracking.

Civics Group Name: 6C35
Civics Tutor: Mr Tan
Chairperson: Steve Lim

Register Number	Student Name	Vaccine brand name	CCA Name	CCA Teacher
11	Steve Lim	Moderna	Choir	Mr Lee
4	Melody Tan	Moderna	Choir	Mr Lee
19	Paul Chang	Pfizer–BioNTech	Soccer	Mr Chan
3	Grace Lee	Pfizer–BioNTech	Dance	Ms Deepa
20	Alison Chong	Not Vaccinated	Infocomm Club	Mrs Ho
12	Vincent Eng	Moderna	Soccer	Mr Chan

(a) Derive a set of tables to show the above data in first, second and third normal form. [6]

(b) Draw an ER diagram for a normalised database design. [2]

(c) Using examples in the above context, explain the significance of the following terms:

i) primary key [2]

ii) foreign key [2]

(d) With reference to the above context, describe or suggest a scenario where a NoSQL database would be more appropriate. [1]

6 Long-distance optical fibre lines and submarine cables which are a vital part of the global internet infrastructure are vulnerable to solar superstorms which happen once in a century. The last solar superstrom was in 1921.

(a) State and explain why websites would or would not be accessible by web browsers if a solar superstorm shuts down all DNS servers. [2]

(b) Explain *packet switching* and its importance in ensuring global internet connectivity when some parts of the earth are hit by a solar superstorm. [4]

An international company based in many countries updates its network structure to ensure Internet connectivity during solar superstorms.

Employees can now access files on a shared virtual space which is made up of 3 servers located in the United States, Europe, and Asia. All servers hold identical information (any changes made on one server would update the other servers immediately) so even if one server is affected by the solar superstorm, employees still can access their files on the other servers.

Employees must be connected physically to the company's intranet network at each country's office to access the virtual space.

(c) Draw a network diagram of the above configuration and label the LAN, internet, router(s), WAN link(s), intranet, severs, and employee laptops. [5]

(d) Why is version control vital when employees from different countries work in teams? [1]

7 A divide and conquer approach is used by merge sort to successively divide a list into half, forming two sublists, until each sublist is of length 1. The sublists are then sorted and merged into larger sublists until they are recombined into a single sorted list. An algorithm for merge sort to perform an ascending sort is given below. It will be used to sort large or small data sizes.

```
PROCEDURE mergesort(mergelist : ARRAY)

    IF LENGTH(mergelist) > 1 THEN

        mid ← LENGTH(mergelist) DIV 2

        FOR index ← 0 TO (mid - 1)
            lefthalf[index] ← mergelist[index]
        NEXT index

        right_len ← LENGTH(mergelist) - mid

        FOR index ← 0  TO (right_len - 1)
            righthalf[index] ← mergelist[right_len + index]
        NEXT index

        mergesort(lefthalf)
        mergesort(righthalf)

        i ← 0
        j ← 0
        k ← 0
        WHILE i < LENGTH(lefthalf) AND j < LENGTH(righthalf)
            IF lefthalf[i] > righthalf[j] THEN
                mergelist[k] ← lefthalf[i]
                i ← i + 1
            ELSE
                mergelist[k] ← righthalf[j]
                j ← j + 1
            ENDIF
            k ← k + 1
        ENDWHILE

        WHILE i < LENGTH(lefthalf)
            mergelist[k] ← lefthalf[i]
            i ← i + 1
            k ← k + 1
        ENDWHILE

        WHILE j < LENGTH(righthalf)
            mergelist(k] ← righthalf[j]
            j ← j + 1
            k ← k + 1
        ENDWHILE
    ENDIF
ENDPROCEDURE
```

(a) The following array of numbers is to be sorted using `mergesort`:

```
mergelist = [2, 4, 2, 8, 2, 8, 9, 1, 3]
```

What are the first two lists to be merged? [1]

(b) Explain what a logic error is, give the line number for the logic error in the above code, and rewrite the line correctly. [2]

The procedure `sorting_proc` uses an optimised bubble sort to sort an array `input_array` in an ascending order. It is used within `modified_mergesort` which is a modified version of `mergesort`.

```
01  PROCEDURE sorting_proc(input_array : ARRAY)
02
03        length ← LENGTH(input_array)
04
05        REPEAT
06              swapped ← FALSE
07
08              FOR curr_elem_index ← 1 to length - 1
09
10                    IF  input_array [curr_elem_index - 1] > input_array
                    [curr_elem_index] THEN
11                          SWAP (input_array [curr_elem_index - 1],
                          input_array [curr_elem_index])
12                          swapped ← TRUE
13                    ENDIF
14
15              ENDFOR
16
17              length ← length - 1
18
19        UNTIL NOT swapped
20
21  ENDPROCEDURE
```

```
PROCEDURE modified_mergesort(mergelist : ARRAY)

    IF LENGTH(mergelist) > 1 THEN

        IF LENGTH(mergelist) < 5 THEN
            sorting_proc(mergelist)
            RETURN

        ELSE

            mid ← LENGTH(mergelist) DIV 2

            FOR index ← 0 TO (mid - 1)
                lefthalf[index] ← mergelist[index]
            NEXT index

            right_len ← LENGTH(mergelist) - mid

            FOR index ← 0  TO (right_len - 1)
                righthalf[index] ← mergelist[right_len + index]
            NEXT index

            mergesort(lefthalf)
            mergesort(righthalf)

            i ← 0
            j ← 0
            k ← 0
            WHILE i < LENGTH(lefthalf) AND j < LENGTH(righthalf)
                IF lefthalf[i] > righthalf[j] THEN
                    mergelist[k] ← lefthalf[i]
                    i ← i + 1
                ELSE
                    mergelist[k] ← righthalf[j]
                    j ← j + 1
                ENDIF
                k ← k + 1
            ENDWHILE

            WHILE i < LENGTH(lefthalf)
                mergelist[k] ← lefthalf[i]
                i ← i + 1
                k ← k + 1
            ENDWHILE

            WHILE j < LENGTH(righthalf)
                mergelist(k) ← righthalf[j]
                j ← j + 1
                k ← k + 1
            ENDWHILE
        ENDIF
    ENDIF
ENDPROCEDURE
```

(c) Would the above modification of `mergesort` improve the algorithm's overall efficiency? Support your answer with a description on how and explanation on why its efficiencyis affected. [4]

The procedure `insertionSort` is an algorithm which uses insertion sort.

```
01  PROCEDURE insertionSort(input_array: ARRAY)
02
03        current_elem_index ← 0
04
05        REPEAT
06              current_elem_index ← current_elem_index + 1
07              compared_item_index ← -1
08              swapped ← FALSE
09
10              REPEAT
11                    compared_item_index ← compared_item_index + 1
12
13                    IF input_array[current_elem_index] <
              input_array[compared_item_index] THEN
14
15                          temp ← input_array[current_elem_index]
16
17                          the value of each element of input_array from
                    compared_item_index to (current_elem_index – 1) is
                    sequentially assigned to each element of input_array
                    from (compared_item_index + 1) to current_elem_index
18
19                          input_array[compared_item_index] ← temp
20
21                          swapped ← TRUE
22                    ENDIF
23
24              UNTIL swapped ← TRUE
25
26        UNTIL current_elem_index = LENGTH(input_array) - 1
27
28  ENDPROCEDURE
```

(d) Modify `insertionSort` and `sorting_proc` to count and store the number of comparisons made in a variable named `comparisons.` Instead of copying all the pseudocode statements, state the line number(s) you want to modify or insert any pseudocode at, followed by the pseudocode statement(s) to be added/modified. [3]

(e) Trace the modified algorithms `insertionSort` and `sorting_proc` for the array `[5, 2, 3, 4]` showing the value of all variables for each step by completing the following tables. [7]

Trace table for `insertionSort`:

current_elem_index	compared_item_index	comparisons	input_array	swapped
1	0	1	[2,5,3,4]	TRUE
2	0	2	[2,5,3,4]	FALSE
...	...	...	...	...

Trace table for `sorting_proc`:

`curr_elem_index`	`comparisons`	`input_array`	`arr_length`	`swapped`
1	1	[2,3,5,4]	4	TRUE
2	2	[2,3,5,4]	4	TRUE
...	...	...	...	...

(f) In the context of `mergesort`, suggest scenario(s) where using the current optimised bubble sort algorithm for `sorting_proc` would be better than using the `insertionSort` algorithm above. Support your answer by designing 3 test cases (normal and boundary) and comparing the number of `comparisons` made by each algorithm for each test case. Display your output. [7]