Jurong Pioneer Junior College

2022 JC 2 H2 Computing (Syllabus: 9569)

Timed – practice (Paper 2)

33

2020 HCI Paper 2 Question 4

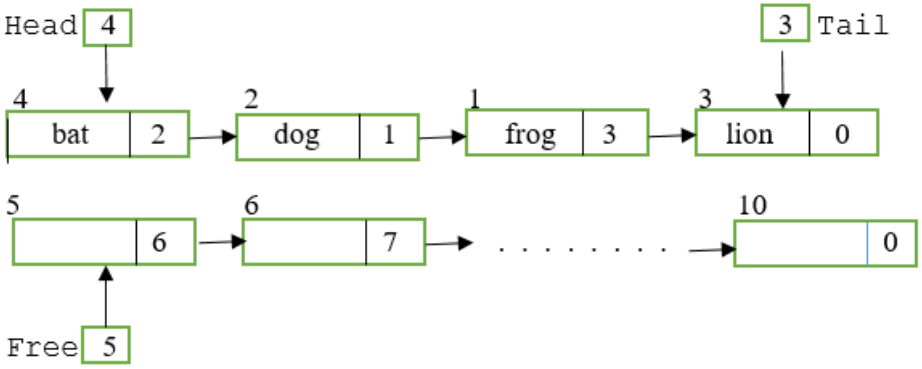
Duration: 1 hour

A programmer is writing a program to manipulate different data structures using Object-Oriented Programming.

The superclass, `LinkedStructure`, will store the following data:

- A linear linked list of data items held in an **array** of size 10. Array index starts at 1.
- Head pointer, pointing to the first element in the linked list
- Tail pointer, pointing to the last element in the linked list
- Free pointer, pointing to the first element in the free list

The diagram shows the linked structure after four items have been added and the unused nodes are linked together.



This superclass has the following methods:

- `Initialise()` method sets up an empty linked list. Should link all nodes to form the free list. Initialise values for head pointer, tail pointer and free pointer
- `Add(item)` appends the parameter into its correct **alphabetical** order in the linked list.
- `Remove(item)` removes the parameter from the ordered linked list
- `Display()` displays the data items in the linked list in alphabetical order
- `PrintStructure()` displays the current state of all pointers and the array contents in index order
- `IsEmpty()` tests for empty linked list
- `IsFull()` tests for no unused nodes

The superclass is used to implement a linear queue.

The subclass `Queue` has the following methods:

- `Add(item)` appends the parameter to the queue and overrides the `LinkedStructure` add method .
- `Remove()` returns and removes the next item in the queue
- `Display()` method should display the queue contents in order (e.g. the earliest added item first) and should override the `LinkedStructure` display method.

Each method updates its appropriate pointers, and produces suitable errors (or returns different values) to indicate if the actions are not possible, e.g. if the structure is empty.

For each sub-task, add a comment statement, at the beginning of the code using the hash symbol "#", to indicate the sub-task the program code belongs to, for example:

```
In [1]: #Task 3.1
        Program code

        Output:
```

Task 4.1

Write program code for the superclass `LinkedStructure`. [20]

Task 4.2

Write program code to:

- create a `LinkedStructure` object
- add the following three data items in the order shown to the ordered linked list:
  `Japan, Singapore, China`
- output all pointers and array contents using the `PrintStructure()` method after adding the items
- output the current contents of the linked list using the `Display()` method
- remove two data items China, Japan in that order from the linked list
- output all pointers and array contents using the `PrintStructure()` method after the removal of the items. [5]

Task 4.3

Write program code for the subclass `Queue`.

Use appropriate inheritance and polymorphism in your designs. [5]

Task 4.4

The file `QUEUE.TXT` stores data to test your program.

Write program code to:

- create a new queue and add the data in the file `QUEUE.TXT` to the queue
- output the current contents of the queue
- remove and output two items from the queue
- output all pointers and the array contents of the queue after the removal of the items.

All outputs should have appropriate messages to indicate what they are showing. [3]

Download your program code and output for the entire **Task 4** as

`TASK4_<your_name>_<centre number>_<index number>.ipynb`