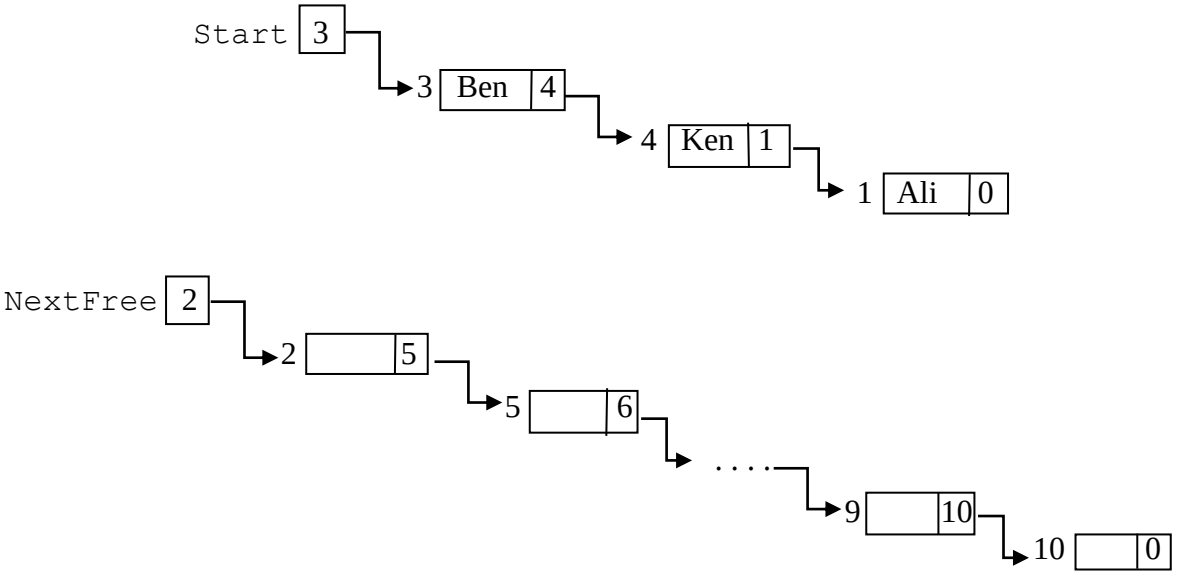
3. A linked list Abstract Data Type (ADT) has the following operations defined:

- `Constructor()` -- creates an empty linked list;
- `Insert(item, p)` -- inserts new value, `item`, into linked list so that it is at position `p` in the linked list. Assume that the linked list contains at least `(p - 1)` items before the insertion.
- `Delete(p)` -- deletes the item at position `p` in the linked list;
- `Length()` -- returns the number of items in the linked list;
- `IsEmpty()` -- returns true if linked list is empty;
- `IsFull()` – returns true if linked list is full;

The linked list is implemented by the use of a collection of nodes that have two parts: the item data and a pointer to the next item in the list. In addition there is a `Start` pointer which points to the first item in the list.

The unused nodes are linked and the first unused node is the position where the next new data item is to be stored. Node removed from the linked list should be returned to `NextFree` list.



The diagram shows the linked list after the following sequence of commands have been executed.

```
Constructor()
Insert('Ali', 1)
Insert('Jack', 1)
Insert('Ben',2)
Delete(1)
Insert('Jane', 2)
Insert('Ken', 3)
Delete(2)
```

The program to implement this ADT will use the classes `ListNode` and `LinkedList` as follows:

ListNode
`Name : STRING` `Pointer : INTEGER`

`Constructor()`
`SetName(Name:STRING)`
`SetPointer(Pointer:INTEGER)`
`GetName():STRING`
`GetPointer():INTEGER`

LinkedList

`Node : ARRAY [1..10] OF ListNode`
`Start : INTEGER`
`NextFree : INTEGER`

`Constructor()`
`Insert (name: STRING, position: INTEGER)`
`Delete (position: INTEGER)`
`Length (): INTEGER`
`IsEmpty(): BOOLEAN`
`IsFull() : BOOLEAN`

Task 3.1
Write program code to define the classes `ListNode` and `LinkedList`.

Evidence 15
Program code for the `ListNode` and `LinkedList` classes. [18]

Task 3.2
A method `Display()` is to be added, which displays the value of `Start`, the value of `NextFree` and the contents of `Node` array in index order. Write program code to implement this method.

Evidence 16
Your program code for Task 3.2. [4]

Task 3.3
Write code to create a `LinkedList` object in the main program. Paste the sequence of commands in `COMMANDS.txt` into your program code. Your program will then call the `Display` method.

Execute your program to test it.

Evidence 17
Screenshot showing the output from running the program in Task 3.3. [2]

A linear queue is implemented using the `LinkedList` class as a super class.

The subclass `Queue` has the following methods:

- `Enqueue(item)` -- inserts `item` at the rear of the queue;
- `Dequeue()` -- deletes the item at the front of the queue;
- `Display()` -- displays the contents of the queue using the format given below.

```
Steven   ← Front
Celine
Tom
Ryan     ← Rear

Number in queue:  4
```

Task 3.4
Write program code for the subclass `Queue`.
Use appropriate inheritance and polymorphism in your design.

Evidence 18
Your program code for Task 3.4. [5]

Task 3.5
Write program code to:

- create a new queue and add the data in the file `NAMES.txt` to the queue
- remove two items from the queue
- display final contents of the queue

Evidence 19
Your program code for Task 3.5. [2]

Evidence 20
Screenshot showing the output from running the program in Task 3.5. [1]