

School of Computer Science and Software Engineering  
 Clayton Campus, Monash University  
**CSE1303 Part A**  
**Summer Semester, 2002**

### Tutorial 4: Linked Structures

Please attempt all starred questions before your tutorial class.

#### Exercise 1.\*

Write a C function `int stackSize(const Stack* stackPtr)` which returns the number of items in a Linked Stack.

#### Exercise 2.\*

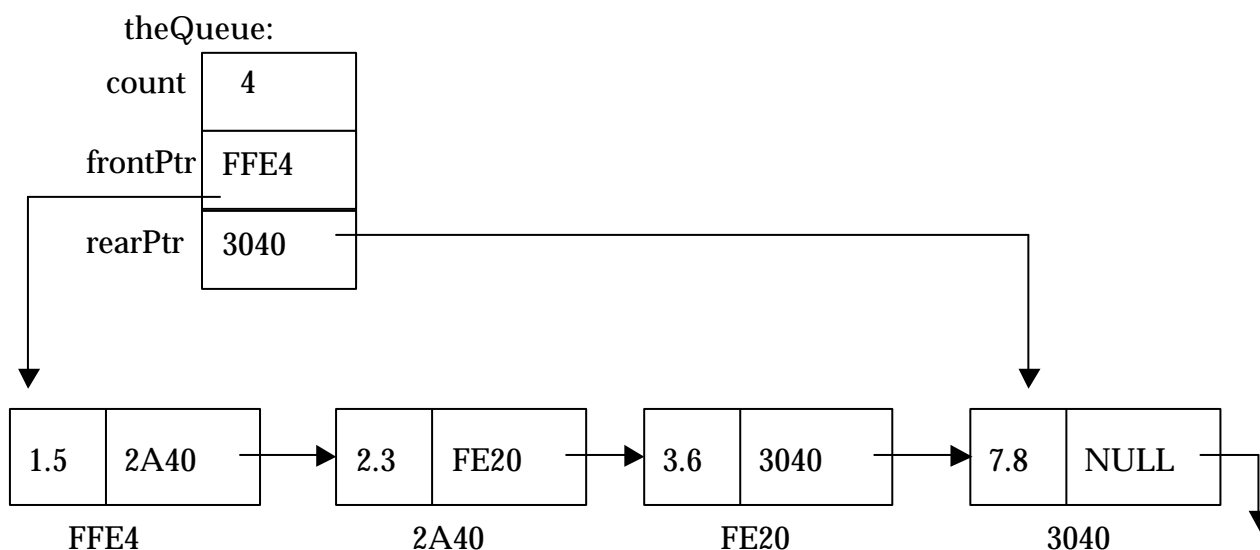
Consider the following code, where `linkQueue.h` is defined as it is in the lecture notes.

```
#include "linkQueue.h"
```

```
float item;
```

```
Queue theQueue;
```

Assume at some stage the Linked Queue associated with `theQueue` can be represented as follows:



Now assume that the following function call is then made:

```
item = serve(&theQueue);
```

- What is now the value of `theQueue.frontPtr` ?
- What is now the value of `theQueue.rearPtr` ?
- What is the value of `item` ?
- What is the value of `(theQueue.frontPtr)->nextPtr` ?

**Exercise 3.\***

Assume that the Linked Queue is defined as it is in the lecture notes to hold items of type **float**. Write a C function **void clearQueue(Queue\* queuePtr)** which deletes all items in the Linked Queue.

**Exercise 4.**

- (a) Change the function **Node\* setPosition(List\* listPtr, int position)** given in the lectures notes so that it returns **NULL** if **position** corresponds to an invalid position in the list.
- (b) Now change the function **void insertItem(List\* listPtr, float item, int position)** given in lectures notes so that it works correctly using the new version of **setPosition** for all possible inputs.

**Exercise 5.\***

Assume the function **char\* strdup(const char\* s)**, (which makes a copy of the string **s** and returns the address of the copy), is defined and suppose

```
struct DoubleLinkNodeRec
{
    char*          str;
    struct DoubleLinkNodeRec* nextPtr;
    struct DoubleLinkNodeRec* previousPtr;
};

typedef struct DoubleLinkNodeRec Node;
```

- (a) Write a C function **Node\* makeNode(const char\* s)** which allocates memory for a new **Node** and also:
- ⊕ returns **NULL** if the memory cannot be allocated,
  - ⊕ assigns the member **str**, of the new node, the address of a copy of **s**, and
  - ⊕ assigns both the members **nextPtr** and **previousPtr**, of the new node, the value **NULL**.
- (b) Write a C function **void killNode(Node\* nodePtr)** which deallocates all the memory associated with the node pointed to by **nodePtr**.

**Exercise 6 (For Prac A04)**

Write a C function that reads a string from the user that could be in any of the following formats, and outputs the correct output as shown below:

<b>number p</b>	<b>Print "Line number to be printed is number - 1"</b>
<b>\$r string</b>	<b>Print "Reading from filename: string"</b>
<b>d number</b>	<b>Print "Deleting line number: number"</b>

Where **number** is an integer, **string** is a character string. The other characters there: **\$r**, **p**, and **d** are to be read literally.

**Additional Exercises**

Kruse et al. 2e

Exercises 3.1: E9 a-b

Exercises 4.6: E2 a-d, E4 a-b, E5 a-c, E6

Exercises 5.1: E1, E2, E3, E4, E5, E6, E8, E9, E10

Exercises 5.3: E2 a-d

Review Questions: 1, 2, 3, 8

Deitel & Deitel 2e

Self Review Exercises 12.1 c, h, 12.2, 12.3, 12.4 a-e

Exercises: 12.6, 12.7