

COMP 1603 - Programming III (Sem 2: 2023_2024) Assignment #2

Linked Lists and Arrays

**Date/Time Due: Sun 10th March 2024 by
midnight**

In this assignment you would be manipulating linked lists that are stored in an array. Each node of a linked list stores a data field (integer) and a next field with the usual meaning. The declaration of a *Node* is shown below.

```
struct Node {
    int data;
    Node * next;
};
```

Data File Information

The data file name is given below. There are 5 lists in all for this assignment and each list's data is stored on a separate line (terminated by -1) and all list data are terminated by -999. See below.

After -999, there is a set of commands that are to be processed using the array of pointers to linked lists. In summary, you are required to store the linked list data in locations of an array of pointers, then process the list of commands.

Further descriptions would appear later.

[Name of file: 'asg2.txt']

The data file contents follow.

| | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|
| 1 | 1 | 2 | 3 | 5 | -1 | | | |
| 2 | 2 | 2 | 3 | 4 | 5 | -1 | | |
| 3 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | -1 |
| 4 | 1 | 2 | 3 | 4 | 5 | | -1 | |
| 5 | 23 | 20 | 18 | 19 | 14 | 12 | -1 | |

-999

G 2

G 3

X 4

A 4 6

A 4 7

A 4 9

A 1 9

P 1

P 4

Z

D 1 5

D 5 23

D 5 88

P 1

P 5

Z

H 1 90

H 2 95
 M 1 2 3
 P 3
 Z
 \$

Further Notes on Data File etc.

In the first set of data to be stored (terminated by -999), each line has a number appearing first and this represents the list number and the numbers after the list number represent that list data (except for -1, the terminator). List number 1 should be stored in location 1 of the array, list number 2 should be stored in location 2 of the array etc. Examples:

Example 1

1 1 2 3 5 1 -1

1 represents the list number and should be stored in location 1 of the array and 1 2 3 5 1 represents the list data. -1 signals the end of data flag that means end of data for that line. -1 is not stored.

The storage of the **list 1** data is illustrated in Figure 1.

Example 2

2 2 2 3 4 5 -1

2 represents the list number and should be stored in location 2 of the array and 2 2 3 4 5 represents the list data. -1 signals the end of data flag. -1 is not stored.

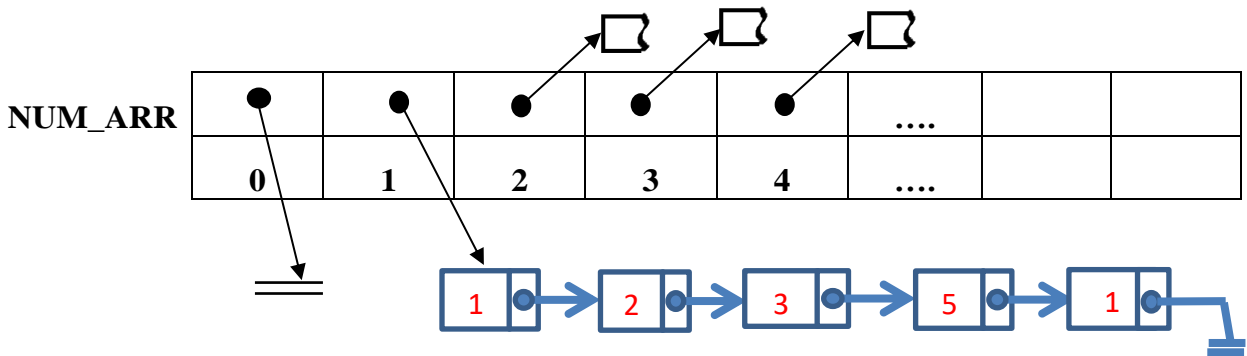



The storage of the **list 2** data is not fully illustrated in Figure 1 but the symbol represents what this list should be. (it is a list abbreviation symbol)

SOME PROCESSING AND STORAGE REQUIREMENTS

1. Diagram of the array of linked lists that is used for storing the data. The name of the array is **NUM_ARR**

Figure 1



2.  This symbol means that there is a linked list or NULL in this location but instead a of a list being drawn, this symbol (list abbreviation) is shown. Only location 1 has a full drawing of a linked list for illustration purposes.
3. means the ideas explained for earlier locations may repeat.
4. Location zero is used to store the number of lists that NUM_ARR contains. Store this counter in the data field of a single node at location 0. The counter represents the number of lists stored from the data file and should be 5 eventually, for 5 lists. This counter would be zero initially, however. 5 would be the counter as there are 5 lists but if the assignment ideas are expanded later on, the lists can grow and so would the counter. **If a full linked list is deleted later on, do not change the counter.**
5. **Processing: Read all the data terminated by -999 and store in the array of linked list pointers, NUM_ARR.**

Commands

After the -999 in the first set of data, commands would appear from the line after, one per line. A single character '\$' terminates this last set of data. Read the data and process the commands and give the relevant output to the screen. Commands are as follows:

G

G <listnum> means find and print the largest integer in list *listnum*. Ties do not matter.

G 1 would therefore print 5. A better message is "The largest integer in list 1 is 5".

X <listnum>

Deletes the entire *listnum* list and frees the storage.

X 4 would delete the entire list 4. Assume *listnum* exists. At the end, list 4 should point to NULL. Do not change the main counter in location 0.

A**A <listnum> x**

Creates a node with integer **x** as the data and adds to the end of list *listnum*.

D**D <listnum> x**

Searches *listnum* for integer **x** and deletes the node with **x** if found. If not found, print a message.

Note: Assume there is only ONE instance of **x**. After the instance of **x** is deleted, do not search again. **If by chance a list has duplicates, delete the first instance only.**

P <listnum>

Print the *listnum* list

Z

Print the entire array of pointer data i.e. the contents of all lists in the array. Label appropriately. Only print information for the locations that were used during data file processing. Specifically- locations 0 to 5. Of course, location 0 contains a counter.

H <listnum> x

Creates a node with integer **x** as the data and adds to the head of list *listnum*. Hint: utilize `insertAtHead`.

M**M A B C – where A, B and C are integers.**

Merge list A with list B (A list, then B list) and store the merge at the head of list C.

E.g.

M 1 2 3 where

List 1: 1 2 3 4

List 2: 5 6 7

List 3: 9 10 11

After M 1 2 3

List 3 would have: 1 2 3 4 5 6 7 9 10 11

Assume that for the **M** command, valid lists would be present, so the merge is possible.

.....

Suggestion for command processing

A suggestion is to have a command router or command processor function that handles all the commands. A sample is provided below **but is not compulsory**.

```
void commandRouter (Node *NUM_ARR[], char command, int param1, int param2, int
param3 ) {
```

```
//coded by Dr. M. Hosein
```

```
    switch (command) {
```

```

case 'G': int largest;

        largest=findLargest(NUM_ARR[param1]);

        cout<<"Largest in list "<<param1<<" is "<<largest<<endl;

        break;

```

Submission: Upload a file containing your **source code** to My Elearning by the deadline. **Do not**
upload the data file

Upload one file only.

Name your source file your first name initial + last name+ID.

e.g. *MSlater816031051* for Marcus Slater. Ensure that you place your name, student ID, course code, and assignment number in documentation at the top of your file.

There is a penalty for prompting.

Queries: After the assignment is marked, queries can only be made on the exact files submitted. It is your responsibility to ensure that you submitted the correct files. Queries are to be made within one week of the release of the assignment results.

Mark Scheme (Once all instructions followed)

| | |
|--|--|
| Stored data as instructed | [5] {If -1 stored, award 3 marks only} |
| 20 commands (except M) work, 2 each | [40] {If they almost work, 1 mark may be awarded per command} |
| M command | [10] |
| Output well-presented and neat | [2] |
| Good attempt/pointers usage | [3] |
| TOTAL | [60] |

Mark scheme override:

If weak answer, syntax or logic errors, the above scheme is overridden with a maximum mark of 10 marks.

Penalties to be determined later during marking:

- penalty for prompting
- penalty for obscure code
- output not listed in the order of commands

Note-No extremely advanced code is allowed. Class code etc. and code along those lines must be used. E.g. recursion is now allowed.