Singly Linked Lists

Academic Honesty

The assignment is individual work. Students are allowed to consult books and online resources but must acknowledge the resources used in the report.

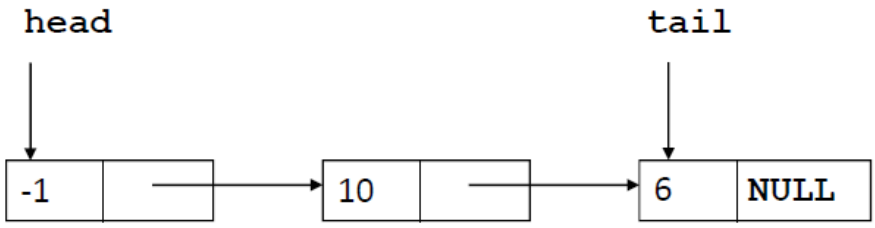
We use MOSS to detect software plagiarism.

Sign the Academic Honesty Pledge in the report.

1. Specification

In this implementation, the singly linked list stores non-negative integers. A dummy node with the data value -1 is used to simplify insertions and deletions. See the diagram below. Variables `head` and `tail` are global variables, with `head` always pointing to the dummy node at the front of the list, and `tail` pointing to the last element at the rear of the list.

Write a C program to implement the search, insertion and deletion operations as described below.



2. Implementation

- The program to be submitted is named `list.c`. Use the given template `list.c` and fill in your code. **Submit file `list.c`.**
- You are also given a file named `listMain.c` to test your code. Do not submit file `listMain.c`.
- Implement the following functions: `insertFirst()`, `insertLast()`, `removeFirst()`, and `search()`. See file `list.c` for their specifications.
- Function `insertFirst()`: The new element is to be inserted at the front of the list, right after the dummy node. If a new node cannot be created (e.g., insufficient memory), the function calls function `prtError()` to display an error message and returns NULL. Otherwise, it returns the pointer to the newly created node.

- Function `insertLast()`: The new element is to be inserted at the end of the list. If a new node cannot be created (e.g., insufficient memory), the function calls function `prtError()` to display an error message and returns NULL . Otherwise, it returns the pointer to the newly created node.
- Function `removeFirst()`: If the list is empty (i.e., no element other than the dummy node), the function calls function `prtError()` to display an error message and returns -1. Otherwise, it removes the first element (i.e., the node right behind the dummy node) and returns the data (integer) of the removed node.
- Function `search()`: If there is an element containing non-negative integer `k` then return the pointer to that element. Otherwise, return NULL. If there is more than one element containing `k`, return the pointer to the first encountered element.
- Assume that all inputs are non-negative integers and there can be duplicates.
- You may define your own variables inside the above functions.
- In file `list.c` you are given these utility functions: `init(), getFirst(), getLast(), prtError()`and `prtList()`. DO NOT modify these functions.
- Do not modify the function or structure definitions in file `list.c` or `list.h`.
- To compile the C files, use the following commands, then run the executable file `listMain`.

```
gcc –Wall –c list.c

gcc –Wall –c listMain.c

gcc list.o listMain.o –o listMain
```

- See file `listMain_out.txt` for the output from running programs `list.c` and `listMain.c`.
- You should create your own `main()` programs following the template `listMain.c` to test each function separately (unit testing) from the simplest case to complex cases.

Important Notes

- Do not use any C library function except malloc(), calloc(), and free(). Do not add any other header file to your C file(s).
- The functions (algorithms) must be the most efficient in terms of running time and memory usage.
- **Submit file `list.c`.** Complete the header of the C file with your student and contact information. Include sufficient comments in your code to facilitate code inspection.

- **Submit a report in PDF** with following information: references (sources); error conditions and actions taken; algorithm; running time of the function (algorithm) and an explanation. See the template a4report.docx for an example.
- Your code will be graded automatically by a script, so make sure to strictly follow the specifications.
- Do not use any output statements (for example, printf) in your code, or they will mess up the grading scripts.
- C compilers are machine-dependent, and programs may behave differently on different systems. C assignments will be graded on an EECS server from the command line by a script. To make sure your code passes these tests, it should be tested on an EECS server (for example, red.eecs.yorku.ca) before the final submission.
- We will use more test cases for grading.