SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 125 - Introduction to Computing Science and Programming II - Fall 2021

Lab 8. Linked List

November 3

Quick Recap - Linked List

- Chain of separate elements
- Head points to the first element
- Tail points to the last element
- Each element has a data part which contains value and a pointer which points to the next node in the list

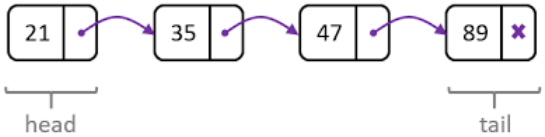

Fig1: Linked List
Source: 101computing

Quick Recap - Linked List Operations

- LL_add_to_head(LL_t *list, int value): Add element to the head of the list
- LL_add_to_tail(LL_t *list, int value): Add element to the tail of the list
- LL_remove_from_head(LL_t *list): Remove element from the head of the list
- LL_size(const LL_t *list): Return the size of the list
- LL_print(const LL_t *list): Prints all elements of the list from head to tail
- LLnode_free(node_t *node): Frees memory used by the node
- LL_free(LL_t *list): Frees memory used by the list

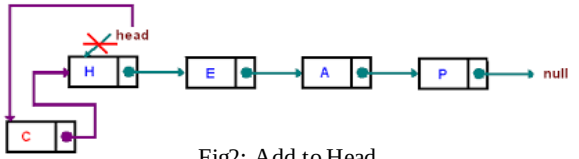

Fig2: Add to Head

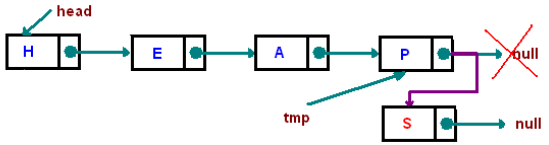

Fig3: Add to Tail

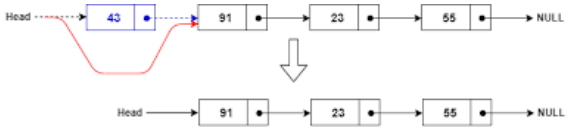

Fig4: Remove from Head

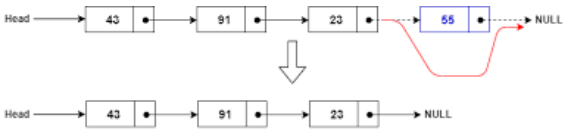

Fig5: Remove from Tail

Exercise

SFU

- Read and understand the functions defined in LL.c

- Implement the functions:
  - LL_remove_from_tail(): removes element from tail of the list
  - LL_print_reverse(): prints list elements in reverse order. Try doing it in O(N) time.
  - to_array(): gets a linked list and creates an array with same values
  - array_to_list(): gets an array and creates a linked list
  - are_equal(): check if two linked lists are equal (equal length and same values in order)

- Add more test cases to test the functions you implement

Steps to compile code

SFU

- Unzip and open the directory in VSCode
- > make
- > ./driver_LL