CT331 Assignment 1

Procedural Programming with C

Due Date: TBC

Introduction

Please submit a single pdf file (only pdf files are accepted) clearly showing the following:

1. The assignment name ("CT331 Assignment 1")
2. Each question title ("Question 1"), in order, displayed clearly above your answer.
3. For each question:
    a. A cropped screenshot of any command line output.
    b. A copy of your code (not a screenshot) – you should only submit the code you have written – NOT the provided code.
    c. Any comment you may have, or whatever textual answer the question requires.

Notes:

The example code is available with this assignment

Questions 2 and 3 of this assignment build upon the Linked List example presented in lectures. Do NOT submit my code as part of your submission.

Please work with the example code, including the function names, data types and file structure as given.

The assignment is marked out of 30, and represents 10% of the final course grade.

Question 1

(A) Write code that creates variables of the following types and prints out their size using sizeof(): [2.5 marks]

- int
- int*
- long
- double *
- char **

(B) Comment on your results.
[2.5 marks]

Question 2

Based on the example linked list code, extend that code to implement the following functions. You will have to update the linkedList.h file and the linkedList.c file.
[2 marks each]

- int length(listElement* list)
  - Returns the number of elements in a linked list.
- void push(listElement** list, char* data, size_t size) ○ Push a new element onto the head of a list. ○ Update the list reference using side effects.
  - (See: swap() from lecture 3)
- listElement* pop(listElement** list)
  - Pop an element from the head of a list.
  - Update the list reference using side effects.

NOTE: We have a linked list stack!

- void enqueue(listElement** list, char* data, size_t size);
  - Enqueue a new element onto the head of the list.
  - Update the list reference using side effects.
- listElement* dequeue(listElement* list);
  - Dequeue an element from the tail of the list.

NOTE: We have a linked list queue!

Question 3

The linked list currently only accepts char* data. Create the files genericLinkedList.h and genericLinkedList.c files, extend the linked list to accept any data type (Hint: Use void* to create a pointer to any data type.)

[10 marks total]

- Ensure all functions from Question 2 work accordingly.
- Traverse() will not be able to format the data (%d, %s, %ld etc...) so…
    - Update the struct to store a function pointer. The function should print out a specific data type (like printStr() will print a string, and printInt() will print an integer) Eg:

      ```
      void printChar(void* data){ printf("%c\n",
            *(char*)data);
      }
      ```

    - When traverse is called, use the function pointer with the element's data to print it out.
      Eg:

      ```
      element->printFunction(element->data);
      ```

Question 4

- Comment on the memory and processing required to TRAVERSE a linked list in reverse (tail to head). [2.5 marks]
- How could the structure of a linked list be changed to make this less intensive? [2.5 marks]