Sample Final Exam

COMP-1410 Introduction to Algorithms & Programming II

School of Computer Science
University of Windsor

Ordered Doubly Linked List: You have been provided with an incomplete code for an Ordered Dynamic Doubly Linked List of strings. Your task is to complete the code by adding some functionalities to the program. Before describing the tasks, read the following initial notes for this question:

Initial Notes:

- This is a doubly linked list. This means
  - Every node has two links one to the next node and one to the previous node.
  - The previous link of the first node is NULL.
  - The next link of the last node is NULL
- You must not only keep a pointer to the beginning of the linked list, for instance **start**, but also keep a pointer to the last node of the linked list, for instance **end**.

Tasks:

- **Task 1:** Develop the function **`printList`**, that will the nodes data from the beginning of the list.
- **Task 2:** Develop the function **`isEmpty`**, that will check if the list is empty or not.
- **Task 3:** Develop the function **`setData`**, that will set the node data and links.
- **Task 4:** Develop the function **`insert`**, that will insert a new node into the doubly inked list. Note that the link list should be in ascending order all the times. You should also take care of the starting and ending pointers as well as the next and previous links for all the nodes affected.
- **Task 5:** Develop the function **`delete`**, that will delete the node with data equal to the given value. If it was successful, it will return the data (char), and otherwise will return '\0'. You should take care of the starting and ending pointers as well as the next and previous links for all the nodes affected.
- **Task 6:** Develop the function **`printListReverse`**, that will print the elements of the linked list from the end, i.e. in descending order.
- **Task 7:** Develop the function **`readData`**, that will read some strings from the input file, "input.txt" and inserts them to the list.
- **Task 8:** Develop the function **`writeData`**, that will store list nodes' data to an output file, "output.txt".

*** Note**: **DO NOT ALTER/DELETE** any part of the existing code. **ONLY complete the incomplete parts**.

Sample execution of the program:

```
An empty ordered Doubly-LinkedList created.
*****************************************
Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 1
Enter a name (Maximum 14 characters): David
The list is:
NULL <--> David <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 1
Enter a name (Maximum 14 characters): Mary
The list is:
NULL <--> David <--> Mary <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 1
Enter a name (Maximum 14 characters): Carl
The list is:
NULL <--> Carl <--> David <--> Mary <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
```

```
   8 to end.
? 1
Enter a name (Maximum 14 characters): Jane
The list is:
NULL <--> Carl <--> David <--> Jane <--> Mary <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 2
Enter character to be deleted: David
David deleted.
The list is:
NULL <--> Carl <--> Jane <--> Mary <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 4
The list in reverse order is:
NULL <--> Mary <--> Jane <--> Carl <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 5
Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 3
```

```
The list is:
NULL <--> Albert <--> Alice <--> Bernard <--> Carl <--> Charlie <--> Jane <--> Mary
<--> Paul <--> Rose <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 6
List stored to the output file.
Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 7
List is empty.

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 Read some words from input file and insert them to the list.
   6 Write the list into the output file.
   7 Empty the list.
   8 to end.
? 8
End of run.
```

```
// Ordered Doubly-LinkedList
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Node_struct {
                                        // String with length of 15
                                        // Link to the next node
                                        // Link to the previous node
} Node;

// Two global pointer variables to keep the start and end of the list, initialized with
NULL.



// Selection menu (This function has already been developed)
void menu(void);
/*************
 * Develop this function at the bottom of this program
 * Print the list.
 */
void printList();
/*************
 * Develop this function at the bottom of this program
 * Check if the list is empty.
 */
int isEmpty();
/*************
 * Develop this function at the bottom of this program
 * Set the node's data, using a string value.
 */
void setData(Node* node, char* value);
/*************
 * Develop this function at the bottom of this program
 * Insert a new node into the list in its correct location.
 */
void insert(Node* newPtr);
/*************
 * Develop this function at the bottom of this program
 * It recieves a string to delete the very first node equal to it, if found.
 * If found, it returns the string value of the deleted node, Otherwise, returns ""
(empty string).
 */
char* delete(char* value);
/*************
 * Develop this function at the bottom of this program
 * Print the list in the reverse order (from the end of the list to the beginning of
the list).
 */
void printListReverse();
/*************
 * Read some words from an input file and insert them to the list.
 */
void readData();
/*************
 * Write the list into an output file.
 */
```

```
void writeData();
/*************
 * Develop this function at the bottom of this program
 * Safely make a linked list empty.
 */
void emptyList();

// main function
int main(void) {
    Node* newPtr = NULL;
    char item[15];

    // Creating the starting/ending node.
    start = (Node*)malloc(sizeof(Node));
    setData(start, "#");
    end = start;
    puts("An empty ordered Doubly-LinkedList created.");
    puts("*****************************************");

    menu();
    printf("%s", "? ");
    unsigned int choice; // user's choice
    scanf("%u", &choice);
    fflush(stdin);

    while (choice != 8) {
        switch (choice) {
        case 1:
            /*************
             * Complete this part:
             * 1- Prompt the user to input a character.
             * 2- Read the user's input.
             * 3- Insert the input character into the list.
             * 4- Print the linkedlist.
             */
            printf("%s", "Enter a name (Maximum 14 characters): ");
            fgets(item, 15, stdin);
            if (item[strlen(item) - 1] == '\n')
                item[strlen(item) - 1] = '\0';
            newPtr = (Node*)malloc(sizeof(Node)); // create node
            setData(newPtr, item);
            insert(newPtr); // insert item in list
            printList();
            break;
        case 2:
            /*************
             * Complete this part:
             * 1- If the list is empty, print out a proper message and skip this case.
             *    Otherwise, prompt the user to input a string.
             * 2- Read the user's input.
             * 3- Delete the node that has the input string from the list,
             *    and Print the linkedlist. If not found, show a proper message.
             */
            if (!isEmpty()) {
                printf("%s", "Enter character to be deleted: ");
                fgets(item, 15, stdin);
                if (item[strlen(item) - 1] == '\n')
                    item[strlen(item) - 1] = '\0';
```

```
            char* found = delete(item);
            if (strcmp(found, "") != 0) {
                printf("%s deleted.\n", item);
                printList();
            }
            else
                printf("%s not found.\n\n", item);
        }
        else
            puts("List is empty.\n");
        break;
    case 3:
        /*************
         * Complete this part:
         * 1- If the list is empty, print out a proper message,
         *    Otherwise, print the linkedlist from the BEGINNING of the list.
         */
        if (!isEmpty())
            printList();
        else
            puts("List is empty.\n");
        break;
    case 4:
        /*************
         * Complete this part:
         * 1- If the list is empty, print out a proper message,
         *    Otherwise, print the linkedlist from the END of the list.
         */
        if (!isEmpty())
            printListReverse();
        else
            puts("List is empty.\n");
        break;
    case 5:
        /*************
         * Complete this part:
         * Call the proper fnnction to read from an input file.
         */
        readData();
        break;
    case 6:
        /*************
         * Complete this part:
         * Call the proper fnnction to write the list into a file.
         */
        if (!isEmpty())
            writeData();
        else
            puts("List is empty.\n");
        break;
        // Handling invalid choice
    case 7:
        /*************
         * Complete this part:
         * Call the proper fnnction to empty the list.
         */
        if (!isEmpty())
```

```
                emptyList();
            puts("List is empty.\n");
            break;
        default:
            puts("Invalid choice.\n");
            break;
        } // end switch

        menu();
        printf("%s", "? ");
        scanf("%u", &choice);
        fflush(stdin);
    }
    puts("End of run.");
}

// Display the menu options.
void menu(void) {
    puts("Enter your choice:\n"
         "   1 to insert an element into the list.\n"
         "   2 to delete an element from the list.\n"
         "   3 to print the list from the beginning.\n"
         "   4 to print the list from the end.\n"
         "   5 Read some words from input file and insert them to the list.\n"
         "   6 Write the list into the output file.\n"
         "   7 Empty the list.\n"
         "   8 to end.");
}

// Print the list
void printList() {









}

// Return 1 if the list is empty, 0 otherwise
int isEmpty() {


}

/*************
 * Develop function setData.
 */
void setData(Node* node, char* value) {




}
```

```
/*************
 * Develop function insert.
 */
void insert(Node* newPtr) {









}

/*************
 * Develop function delete.
 */
char* delete(char* value) {
```

```
}

/**************
 * Develop function printListReverse.
 */
void printListReverse() {









}

/**************
 * Develop function readData.
 */
void readData() {









}

/**************
 * Develop function writeData.
 */
void writeData() {
```

```
}

/*************
 * Develop function emptyList.
 */
void emptyList() {









}
```