CCC 204 Data Structures and Algorithms LABORATORY
REPORT :
LAB 7# - Linked List

Samano, Rajel Johann
Information Technology Program, CABECS
Colegio San Agustin – Bacolod

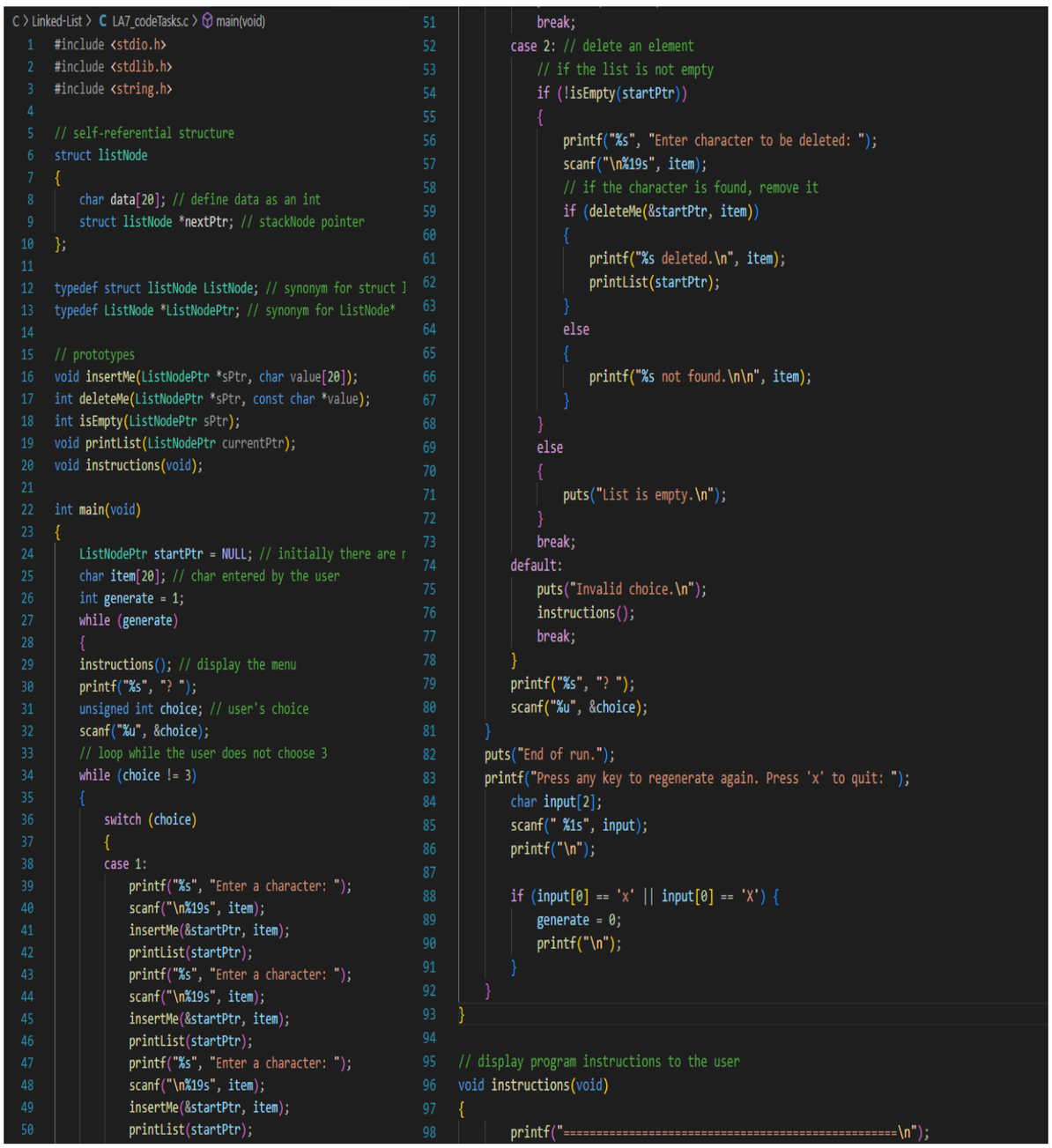
I. INTRODUCTION

For Laboratory Activity Number Seven is to follow the objectives of the exercises and answer questions. The Three main objectives as follows:

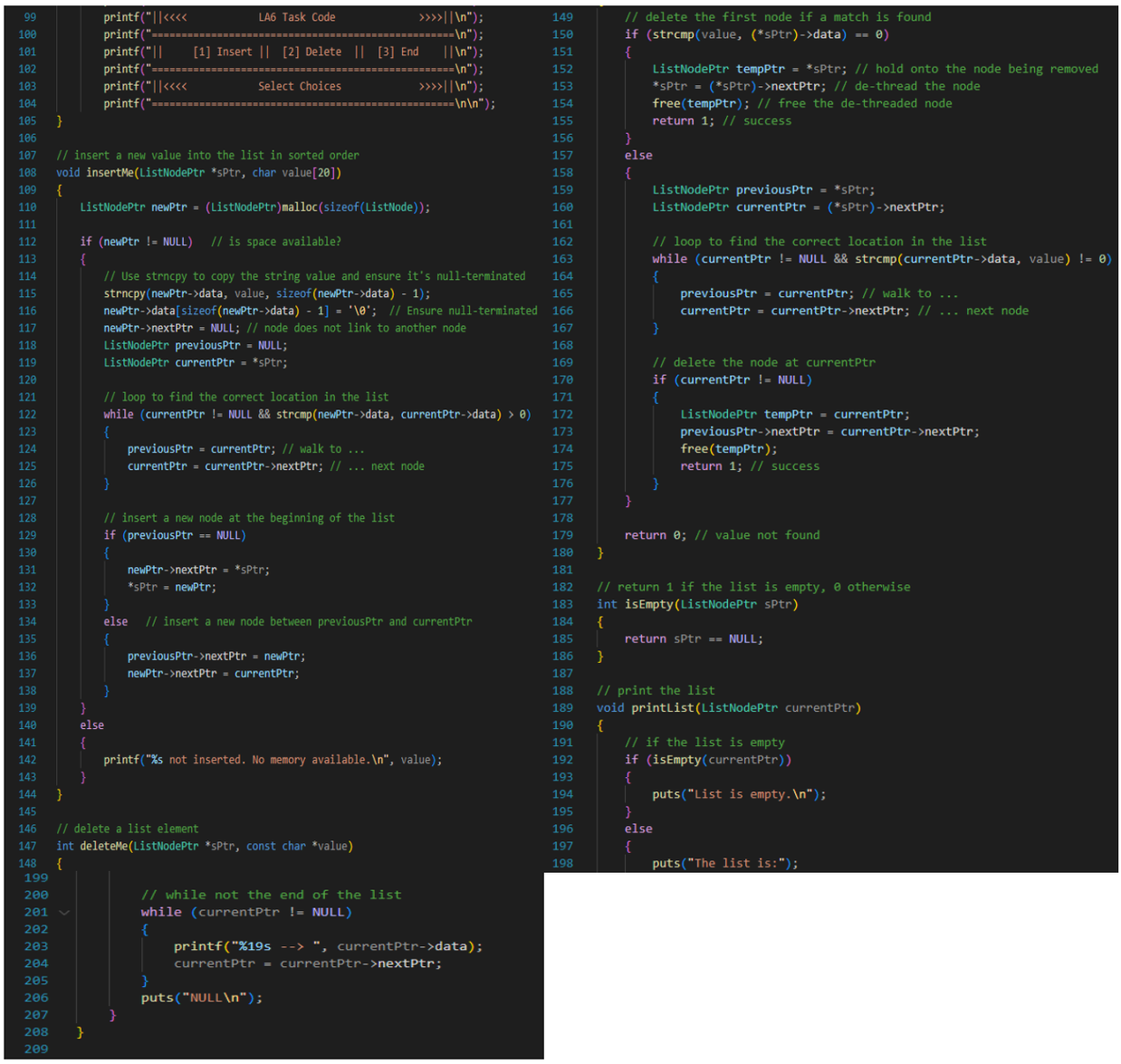
- **Describe linked list**
- **Use self-referential struct for linked list implementation**
- **Perform operations and functions in struct**

II. IMPLEMENTATION / APPROACH

Figure 1-5. LA7_codeTasks.c with output



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// self-referential structure
struct listNode
{
    char data[20]; // define data as an int
    struct listNode *nextPtr; // stackNode pointer
};

typedef struct listNode ListNode; // synonym for struct l
typedef ListNode *ListNodePtr; // synonym for ListNode*

// prototypes
void insertMe(ListNodePtr *sPtr, char value[20]);
int deleteMe(ListNodePtr *sPtr, const char *value);
int isEmpty(ListNodePtr sPtr);
void printList(ListNodePtr currentPtr);
void instructions(void);

int main(void)
{
    ListNodePtr startPtr = NULL; // initially there are r
    char item[20]; // char entered by the user
    int generate = 1;
    while (generate)
    {
    instructions(); // display the menu
    printf("%s", "? ");
    unsigned int choice; // user's choice
    scanf("%u", &choice);
    // loop while the user does not choose 3
    while (choice != 3)
    {
        switch (choice)
        {
        case 1:
            printf("%s", "Enter a character: ");
            scanf("\n%19s", item);
            insertMe(&startPtr, item);
            printList(startPtr);
            printf("%s", "Enter a character: ");
            scanf("\n%19s", item);
            insertMe(&startPtr, item);
            printList(startPtr);
            printf("%s", "Enter a character: ");
            scanf("\n%19s", item);
            insertMe(&startPtr, item);
            printList(startPtr);
            break;
        case 2: // delete an element
            // if the list is not empty
            if (!isEmpty(startPtr))
            {
                printf("%s", "Enter character to be deleted: ");
                scanf("\n%19s", item);
                // if the character is found, remove it
                if (deleteMe(&startPtr, item))
                {
                    printf("%s deleted.\n", item);
                    printList(startPtr);
                }
                else
                {
                    printf("%s not found.\n\n", item);
                }
            }
            else
            {
                puts("List is empty.\n");
            }
            break;
        default:
            puts("Invalid choice.\n");
            instructions();
            break;
        }
        printf("%s", "? ");
        scanf("%u", &choice);
    }
    puts("End of run.");
    printf("Press any key to regenerate again. Press 'x' to quit: ");
        char input[2];
        scanf(" %1s", input);
        printf("\n");

        if (input[0] == 'x' || input[0] == 'X') {
            generate = 0;
            printf("\n");
        }
    }
}

// display program instructions to the user
void instructions(void)
{
        printf("=============================================\n");
```



```
        printf("||<<<<            LA6 Task Code             >>>>||\n");
        printf("=================================================\n");
        printf("||     [1] Insert ||  [2] Delete  ||  [3] End    ||\n");
        printf("=================================================\n");
        printf("||<<<<            Select Choices            >>>>||\n");
        printf("=================================================\n\n");
}

// insert a new value into the list in sorted order
void insertMe(ListNodePtr *sPtr, char value[20])
{
    ListNodePtr newPtr = (ListNodePtr)malloc(sizeof(ListNode));

    if (newPtr != NULL)   // is space available?
    {
        // Use strncpy to copy the string value and ensure it's null-terminated
        strncpy(newPtr->data, value, sizeof(newPtr->data) - 1);
        newPtr->data[sizeof(newPtr->data) - 1] = '\0';  // Ensure null-terminated
        newPtr->nextPtr = NULL; // node does not link to another node
        ListNodePtr previousPtr = NULL;
        ListNodePtr currentPtr = *sPtr;

        // loop to find the correct location in the list
        while (currentPtr != NULL && strcmp(newPtr->data, currentPtr->data) > 0)
        {
            previousPtr = currentPtr; // walk to ...
            currentPtr = currentPtr->nextPtr; // ... next node
        }

        // insert a new node at the beginning of the list
        if (previousPtr == NULL)
        {
            newPtr->nextPtr = *sPtr;
            *sPtr = newPtr;
        }
        else   // insert a new node between previousPtr and currentPtr
        {
            previousPtr->nextPtr = newPtr;
            newPtr->nextPtr = currentPtr;
        }
    }
    else
    {
        printf("%s not inserted. No memory available.\n", value);
    }
}

// delete a list element
int deleteMe(ListNodePtr *sPtr, const char *value)
{
    // delete the first node if a match is found
    if (strcmp(value, (*sPtr)->data) == 0)
    {
        ListNodePtr tempPtr = *sPtr; // hold onto the node being removed
        *sPtr = (*sPtr)->nextPtr; // de-thread the node
        free(tempPtr); // free the de-threaded node
        return 1; // success
    }
    else
    {
        ListNodePtr previousPtr = *sPtr;
        ListNodePtr currentPtr = (*sPtr)->nextPtr;

        // loop to find the correct location in the list
        while (currentPtr != NULL && strcmp(currentPtr->data, value) != 0)
        {
            previousPtr = currentPtr; // walk to ...
            currentPtr = currentPtr->nextPtr; // ... next node
        }

        // delete the node at currentPtr
        if (currentPtr != NULL)
        {
            ListNodePtr tempPtr = currentPtr;
            previousPtr->nextPtr = currentPtr->nextPtr;
            free(tempPtr);
            return 1; // success
        }
    }

    return 0; // value not found
}

// return 1 if the list is empty, 0 otherwise
int isEmpty(ListNodePtr sPtr)
{
    return sPtr == NULL;
}

// print the list
void printList(ListNodePtr currentPtr)
{
    // if the list is empty
    if (isEmpty(currentPtr))
    {
        puts("List is empty.\n");
    }
    else
    {
        puts("The list is:");

            // while not the end of the list
            while (currentPtr != NULL)
            {
                printf("%19s --> ", currentPtr->data);
                currentPtr = currentPtr->nextPtr;
            }
            puts("NULL\n");
        }
    }
```

My approach towards the problem was to first change the variable at listNode from int to char as stated in what we are required to do which was to add a char and string type member to the struct which I did but I only add char as string is just a char array. After that I change all occurrences with the word %c to %19s to get the string. After that I made some adjustments to 3 functions the first one I change is the instructions functions as I put my design from my various codes before. The second one I modified was the insertMe function so that it could get the string value from the input of the user then added some strncpy which copies characters from string and strcmp which is used to compare that it is greater than 0. Then deleteMe change it from char to int and changing the returns from value to 1 and 0 as it they would signify the success and failure if you wanted to deleted something from the list and added strcmp.

III. EXPERIMENTAL FINDINGS / DISCUSSIONS

What I found through this is that linked list needs a lot functions, structs and pointers while very hard to understand what I need to change as changing one thing breaks the other and it gave me a very hard time trying to debug and trying to find a solution in google or if it even is close enough to what I need.

IV. CONCLUSIONS

There I conclude that linked list is very hard to do but from what I had found it is useful for insertion and deletion but not so much for searching there is not much I could say about linked list but from what I experienced fidgeting the codes that my teacher provided it need a lot of understanding which I myself no I have not understood it I know maybe a little.

References:

https://stackoverflow.com/questions/2429217/

https://www.geeksforgeeks.org/data-structures/linked-list/

https://stackoverflow.com/questions/33116474/linked-lists-with-strings

https://www.codeproject.com/Questions/1211568/How-can-I-create-a-generic-linked-list-for-storing