Nalongsone Danddank Student ID : 14958950 StarID: jf3893pd

Email: [nalongsone.danddank@my.metrostate.edu](mailto:nalongsone.danddank@my.metrostate.edu)\

**ICS 365. Organization of Programming Languages.**

**Programming Assignment 3**

**Due**: See Syllabus

**Points**: 40

**Problem:** Write a C program that implements a sorted linked list. You have to implement only three operations: insert, that adds an element into the list, print that prints the entire list (this would be

in sorted order), and remove deletes or removes the linked list (including any allocated storage).

The program prompts the user to enter one of the following three requests: insert, print, remove, or exit. insert lets the user add a char to the list, print prints the list, remove gets rid of the current list and frees any memory (hint: lookup free(pointer)), and exit terminates the program. For other requests the program displays an error message.

Some additional rules, cannot exit while there is a none null linked list (have to remove first), can only enter lower case letters ‘a’ to ‘z’.

Below is a sample run of the program.

> ***insert***

enter a char: a

> ***insert***

enter a char: b

> ***print***

a b

> ***insert***

enter a char: c

> ***print***

a b c

> ***insert***

enter a char: b

> ***sort***

Error: unknown request 'sort'

> ***print***

a b b c

> ***remove***

Linked list is now empty

> ***exit***

The ***bold italic*** text in the example above represents the user input.

**Organization:** You have to follow the user interface requirements illustrated in the above example. You must also use structures to store each element of the linked list. Write separate functions for insert, remove and print. You also need to create each node in the list using the **C** library function **malloc** as shown in the example below.

Solution:

#include <stdio.h>

#include <stdlib.h>

// declaration structors Node for linked list.

struct Node {

char \*ch;

struct Node \*next;

};

// decalration structor new node for creation.

struct Node \*newNode(char new\_ch){

struct Node \*new\_node = (struct Node\*)malloc(sizeof(struct Node));

new\_node->ch = new\_ch;

new\_node->next = NULL;

return new\_node;

}

//declaration functions.

void removes(struct Node \*\*);

void insert(struct Node \*\*, struct Node \*);

void print(struct Node \*);

int main(int argc, char \*argv[]) {

struct Node\* head = NULL;

struct Node\* new\_node;

char str[100];

char exitStr[] = "exit";

char insertStr[] = "insert";

char printStr[] = "print";

char removeStr[] = "remove";

char ch;

printf("Enter one of the following commandline: insert, print, remove, or exit.\n");

scanf("%s", str);

getchar();

// loop until user input 'exit'.

while(strcmp(str,exitStr) != 0){

// checking, allow user only input following the requestment key worlds.

if((strcmp(str,insertStr) != 0) && (strcmp(str,printStr) != 0)

&&(strcmp(str,removeStr) != 0)){

printf("Error: unknown request \'%s\'\n", str);

} else if (strcmp(str,insertStr) == 0){

printf("enter a char: ");

ch = getchar();

// checking, user only input low case letter.

if(ch <97 || ch > 122){

printf("Please enter only low case letters 'a' to 'z'.\n");

ch = NULL;

} else {

new\_node = newNode(ch);

insert(&head, new\_node);

}

} else if (strcmp(str,printStr) == 0){

print(head);

} else if (strcmp(str,removeStr) == 0){

removes(&head);

}

scanf("%s", str);

getchar();

}

// free memory allocate.

if(head){

free(head);

head = NULL;

free(new\_node);

new\_node = NULL;

}

return 0;

}

/\* for insert character to the linked list and also sorted.

\* @params head\_node - struct Node

\* new\_node - struct Node

\*/

void insert(struct Node \*\*head\_node, struct Node \*new\_node){

struct Node \*curr\_node;

if(\*head\_node == NULL || (\*head\_node)->ch >= new\_node ->ch){

new\_node->next = \*head\_node;

\*head\_node = new\_node;

} else {

curr\_node = \*head\_node;

while(curr\_node->next != NULL && curr\_node->next->ch < new\_node->ch){

curr\_node = curr\_node->next;

}

new\_node->next = curr\_node->next;

curr\_node->next = new\_node;

}

};

/\* for printing sorted linked list of characters.

\* @params head\_node - struct Node

\*/

void print(struct Node \*head\_node){

if(head\_node == NULL)

printf("Linked list is now empty");

struct Node \*temp = head\_node;

while(temp != NULL){

printf("%c ", temp->ch);

temp = temp->next;

}

printf("\n");

}

/\* remove all characters in linked list and

\* free the linked list memory allocate.

\* @params head\_node - struct Node

\*/

void removes(struct Node \*\*head\_node){

if(\*head\_node){

free(\*head\_node);

\*head\_node = NULL;

}

printf("Linked list is now empty\n");

}