

CPSC 3125, Operating Systems Lab Assignment

Lab 06

1 Instructions

Using the starter code below, implement a linked list. You are given these four functions:

- int menu();
 - void dowhat(int);
 - Node *create_ll(Node *);
 - Node *display(Node *);
- You are to implement these four functions:
- Node *insert_beg(Node *);
 - Node *insert_end(Node *);
 - Node *insert_before(Node *);
 - Node *insert_after(Node *);

I have given you an executable you can use as a guide. This is a strenuous exercise, but you will learn a lot about pointer manipulatio.

2 Starter Code

```

1  /*****
2  * Name: linked_list_lab.c
3  * Author: C data structures
4  * October 23, 2021
5  * Purpose: Write a program to create a linked list and perform insertions and deletions of
6             all cases.
7  * Write functions to sort and finally delete the entire list at once.
8  * Compile with: linked_list_lab.c -o linked_list_lab.exe -Wall
9  * *****/
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <malloc.h>
13
14 struct node
15 {
16     int data;
17     struct node *next;
18 };
19
20 typedef struct node Node;
21
22 //variable declarations
23 int option;
24 Node *start = NULL;
25
26 //function declarations
27 int menu();
28 void dowhat(int);
29
30 Node *create_ll(Node *);
31 Node *display(Node *);
32 Node *insert_beg(Node *);
33 Node *insert_end(Node *);

```

```

34 Node *insert_before(Node *);
35 Node *insert_after(Node *);
36
37 int main()
38 {
39     system("clear");
40     do
41     {
42         option = menu();
43         printf("start == %p\n", start);
44         dowhat(option);
45     }
46     while(option != 13);
47
48     return 0;
49 }
50
51 int menu()
52 {
53     printf("\n\n*****MAIN MENU*****");
54     printf("\n1: Create a list");
55     printf("\n2: Display the list");
56     printf("\n3: Add a node at the beginning");
57     printf("\n4: Add a node at the end");
58     printf("\n5: Add a node before a given node");
59     printf("\n6: Add a node after a given node");
60     printf("\n13: EXIT");
61     int option;
62     printf("\n\nEnter your option: ");
63     scanf("%d", &option);
64
65     return option;
66 }
67
68 void dowhat(int option)
69 {
70     switch(option)
71     {
72         case 1: start = create_ll(start);
73                 printf("\nLINKED LIST CREATED");
74                 break;
75         case 2: start = display(start);
76                 break;
77         case 3: start = insert_beg(start);
78                 break;
79         case 4: start = insert_end(start);
80                 break;
81         case 5: start = insert_before(start);
82                 break;
83         case 6: start = insert_after(start);
84                 break;
85     }
86 }
87
88 Node *create_ll(Node *start)
89 {
90     Node *new_node, *ptr;
91     int num;
92     printf("Enter the data (-1 to end): ");
93     scanf("%d", &num);
94     while(num != -1)
95     {
96         new_node = (Node*) malloc(sizeof(Node));
97         new_node->data = num;
98         if (start == NULL)
99         {
100             // printf("if branch, start == NULL\n");
101             new_node->next = NULL;

```

```

102         start = new_node;
103     }
104     else
105     {
106         //printf(" else _branch, _start == %p\n", start);
107         ptr = start;
108         while(ptr->next!=NULL)
109             ptr = ptr->next;
110         ptr->next = new_node;
111         new_node->next = NULL;
112     }
113     printf(" _Enter _the _data_: _");
114     scanf("%d", &num);
115 }
116 return start;
117 }
118
119 Node *display(Node *start)
120 {
121     Node *ptr;
122     ptr = start;
123     printf("\nLinked _List _\n");
124     printf(" _start->%p\n", start);
125     while(ptr != NULL)
126     {
127         printf(" _%p<-_%d->%p\n", ptr, ptr->data, ptr->next);
128         ptr = ptr->next;
129     }
130     printf(" _\n");
131     return start;
132 }
133
134 Node *insert_beg(Node *start)
135 {
136     //declare a pointer to a ne Node
137     //declare a new integer as a data value
138     printf(" _n_Enter _the _data_: _");
139     //get user input for the data
140     //malloc memory for a new Node
141     //initialize the data member to the new integer
142     //initialize the next member to the start of the list
143     //set start to the address of the new Node
144     printf("in _insert _beg () _new_node _address: _%p, _new_node->data: _%d, _new_node->next: _%p\n", new_node, new_node->data, new_node->next);
145     return start;
146 }
147
148 Node *insert_end(Node *start)
149 {
150     //declare a pointer to a new Node, and a pointer to a iteration node
151     //declare a new integer as a data value
152     printf(" _Enter _the _data_: _");
153     //get user input for the data
154     //malloc memory for a new Node
155     //initialize the data member to the new integer
156     //initialize the next member to NULL (the end of the list)
157     //set the iteration pointer to start
158     while(ptr->next != NULL)
159         ptr = ptr->next;
160     //set the iteration pointer next member to the new Node
161     printf("in _insert _end () _new_node _address: _%p, _new_node->data: _%d, _new_node->next: _%p\n", new_node, new_node->data, new_node->next);
162     return start;
163 }
164
165 Node *insert_before(Node *start)
166 {
167     //declare a pointer to a new Node, a pointer to a iteration node, and a pointer to the

```

```

    "pre" insertion Node
168 //declare a new integer as a data value, and a new integer to hold the value before
    which the new Node is to be inserted
169 printf("\nEnter the data:");
170 //get the data value from the user
171 printf("\nEnter the value before which the data has to be inserted:");
172 //get the "before" value from the user
173 //malloc memory for a new Node
174 //initialize the data member to the new integer
175 //set the iteration pointer to start
176 while(ptr->data != val)
177 {
178     preptr = ptr;
179     ptr = ptr->next;
180 }
181 //set the next member of the pre-pointer to the new Node
182 printf("In insert before() \nnew_node address: %p, new_node->data: %d, new_node->next: %p\n", new_node, new_node->data, new_node->next);
183 return start;
184 }
185
186 Node *insert_after(Node *start)
187 {
188     //declare a pointer to a new Node, a pointer to a iteration node, and a pointer to the
    "pre" insertion Node
189 //declare a new integer as a data value, and a new integer to hold the value after which
    the new Node is to be inserted
190 printf("\nEnter the data:");
191 //get the data value from the user
192 printf("\nEnter the value after which the data has to be inserted:");
193 //get the "after" value from the user
194 //malloc memory for a new Node
195 //initialize the data member to the new integer
196 //set the iteration pointer to start
197 //set the preptr to ptr
198 while(preptr->data != val)
199 {
200     preptr = ptr;
201     ptr = ptr->next;
202 }
203 //set the next member of the pre-pointer to the new Node
204 //set the next member of the new Node to the pointer
205 printf("In insert after() \nnew_node address: %p, new_node->data: %d, new_node->next: %p\n", new_node, new_node->data, new_node->next);
206 return start;
207 }

```

3 Output

See the demonstration in class and the accompanying executable.

4 Lab deliverable

Your deliverable consists of (1) the C source code, and (2) a text document showing the output of the program.