**Data Structures and Algorithms**

**Lab 2**

**Submitted To:**

Mr. Dilshad Sabir

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

**Build and run the program given in Code Listing 1.** **Implement the *“listLength”* function already made in ‘employees.c’.**

/\*\* This file will contains the definition of one element/node (structure) of the linked list

and the functions associated with the list.

\*/

struct employee

{

char name[50];

int age;

float bs;

struct employee \* next;

};

void inEmpRec(struct employee \* emp);

void displayEmpRec(struct employee \* emp);

void printList(struct employee \* emp);

struct employee \* insertEnd(struct employee \* last);

bool isEmpty(struct employee \* emp);

int listLength(struct employee \* emp);

**employee.h**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include"employee.h"

/\*\* This function takes pointer to a node as argument

and then prompts user to input the data for that

employee. The function does not return anything.

\*/

void inEmpRec(struct employee \* emp)

{

printf("\nEnter the name of the employee: ");

fgets(emp->name, 50, stdin);

printf("\nEnter the age of the employee: ");

scanf("%d", &(emp->age));

printf("\nEnter the basic salary of the employee: ");

scanf("%f", &(emp->bs));

printf("\nRecord entered !\n");

}

/\*\* This function takes pointer to a node as argument

and then prints the record on the screen (stdout).

The function does not return anything.

\*/

void displayEmpRec(struct employee \* emp)

{

printf("\nName:\t %s", emp->name);

printf("Age:\t %d\n", emp->age);

printf("Basic Salary:\t %f\n", emp->bs);

}

/\*\* Function to display the list

\*/

void printList(struct employee \* emp)

{

struct employee \* ptr = emp;

printf("\nStart of list: \n");

//start from the beginning

while(ptr != NULL)

{

displayEmpRec(ptr);

ptr = ptr->next;

}

printf("\nEnd of list.\n");

}

/\*\* Insert a new node at the last location. i.e. after the current one.

If the list is currently empty, a new node is created for the head.

\*/

struct employee \* insertEnd(struct employee \* first)

{

struct employee \* temp = first;

///create a new node

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

inEmpRec(new\_node); /// get data for the newly created node.

///point its next pointer to NULL

new\_node->next = NULL;

if(isEmpty(first)) /// if currently the list is empty

{

temp = new\_node;

return(temp);

}

else

{

while((temp->next) != NULL)

temp = temp->next; /// scroll to the end of the list

temp->next = new\_node;

}

return(first);

}

/\*\* Tests if the list is empty. ONLY the head of the list

should be passed to this function.

\*/

bool isEmpty(struct employee \* emp)

{

return (emp == NULL);

}

/\*\* Function to find the length (in number of nodes) of the list.

\*/

int listLength(struct employee \* emp)

{

int length = 0;

struct employee \* current;

for(current = emp; current != NULL; current = current->next)

{

length++;

}

return length;

}

**employee.c**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include "employee.h"

void flush();

int main()

{

int choice;

struct employee \* head = NULL; /// Pointer to the first element (head) of the list

printf("Hello! This program lets you manage your Employees' Database:\n");

while(1)

{

printf("\nWhat do you want to do now?\n");

printf("\n1. Enter a new node to the list.");

printf("\n2. Print the list.");

printf("\n3. Delete the last item from the list.");

printf("\n4. Save the list to a file.");

printf("\n5. Exit the menue.\n");

scanf("%d", &choice);

flush();

switch (choice)

{

case 1: /// Add a new node to the list at the end. Create one if empty.

{

printf("\nAdding a new node: \n\n");

head = insertEnd(head);

break;

}

case 2: /// Print the list.

{

printList(head);

break;

}

case 3: /// Delete the last item from the list

{

printf("\nThis function is not yet implemented\n");

break;

}

case 4: /// Save list to file

{

printf("\nThis function is not yet implemented\n");

break;

}

case 5: /// Exit the program.

{

printf("\nExiting on user request.\n\n");

return(0);

}

}

}

return 0;

}

void flush()

{

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

**main.c**

**Code Listing 1**

**Program:** In this program, already provided code has been compiled and its working has been analyzed. The function ***“listLength”*** has been implemented by calling it in *‘main.c’* and passing first node **‘head’** as argument to the function. As a result; length of list is printed on the console.

1 #include <stdio.h>

2 #include <stdlib.h>

3 #include <stdbool.h>

4

5 #include "employee.h"

6

7 **void** flush();

8

9 **int** main()

10 {

11 **int** choice;

12

13 **struct** employee \* head = NULL; **/// Pointer to the first element (head) of the list**

14 3

15

16 printf("Hello! This program lets you manage your Employees' Database:\n");

17

18 **while**(1)

19 {

20 printf("\nWhat do you want to do now?\n");

21

22 printf("\n1. Enter a new node to the list.");

23 printf("\n2. Print the list.");

24 printf("\n3. Delete the last item from the list.");

25 printf("\n4. Save the list to a file.");

26 printf("\n5. Exit the menue.\n");

27

28

29

30 scanf("%d", &choice);

31 flush();

32 **switch** (choice)

33 {

34 **case** 1: **/// Add a new node to the list at the end. Create one if empty.**

35 {

36 printf("\nAdding a new node: \n\n");

37 head = insertEnd(head);

38 **break**;

39 }

40

41 **case** 2: **/// Print the list.**

42 {

43 printList(head);

44 **break**;

45 }

46

47 **case** 3: **/// Delete the last item from the list**

48 {

49 printf("\nThis function is not yet implemented\n");

50 **break**;

51 }

52

53 **case** 4: **/// Save list to file**

54 {

55 printf("\nThis function is not yet implemented\n");

56 **break**;

57 }

58

59 **case** 5: **/// Exit the program.**

60 {

61 printf("\nExiting on user request.\n\n");

62 **return**(0);

63 }

64 }

65 }

66 **return** 0;

67 }

68

69 **void** flush()

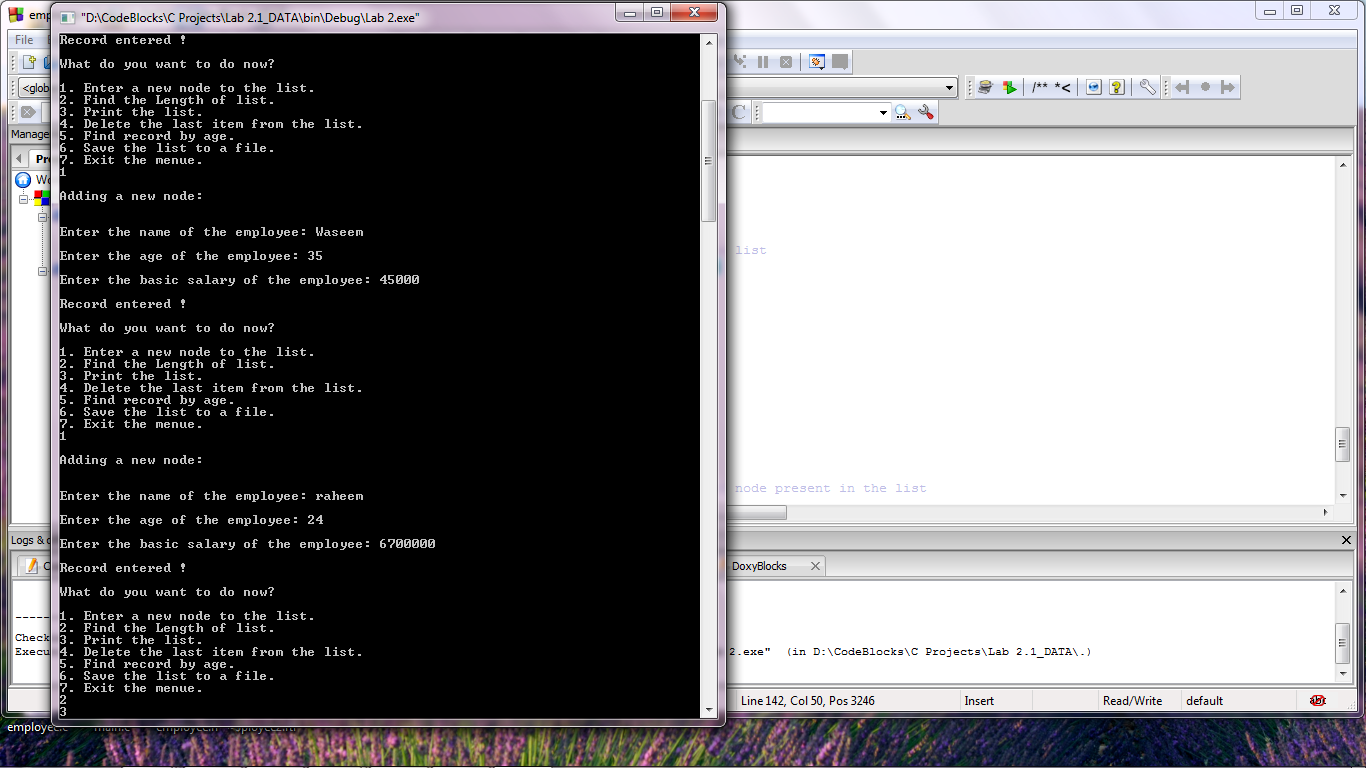
70 {

71 **int** c;

72 **while** ((c = getchar()) != '\n' && c != EOF);

73 }

**Output:**



**Task 2:**

**Find records by age of employees in the linked list.**

**Program:** In this program, a function ***“searchNode”*** takes first node of the linked list and the reference age as its argument. Variables **‘i’** and **‘flag’** are initialized as **1**and **false** respectively. A pointer *‘current’* of type **struct employee** is initialized with the first node of list. Function ***“isEmpty”*** checks if the list is empty otherwise while loop iterates till **‘next’** pointer of *‘current’* doesn’t point to **NULL** and checks if **‘age’** of *‘current’* is equal to *‘key’* i.e. reference age entered by user; **flag** is marked **‘true’** and position of respective node is printed on the console depending upon variable **‘i’**. If **flag** remains 0, a message of required node not found is displayed on the console.

21 **void** searchNode(**struct** employee \* first, **int** key);

**employee.h**

141 //searchNode() will search for a given node in the list

142 **void** searchNode(**struct** employee \* first, **int** key)

143 {

144 **struct** employee \*current = first;

145 **int** i = 1;

146 **bool** flag = **false**;

147

148 //Checks whether list is empty

149 **if**(isEmpty(first))

150 {

151 printf("List is empty \n");

152 }

153 **else** {

154 **while**(current != NULL) {

155 //Compares node to be found with each node present in the list

156 **if**(current->age == key) {

157 flag = **true**;

158 printf("Element %d is present in the list at the position: %d\n",key, i);

159 }

160 i++;

161 current = current->next;

162 }

163 }

164 **if**(flag==0)

165 printf("Element %d is not present in the list\n",key);

166 }

**employee.c**

64 **case** 5: **/// Find record by age.**

65 {

66 printf("\nEnter age of whose record you want to find.\n");

67 scanf("%d",&age\_key);

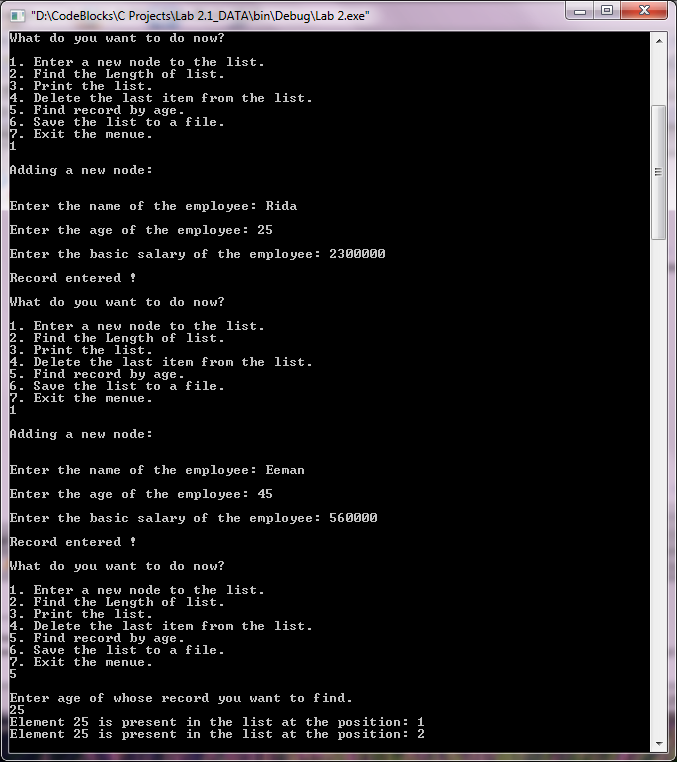
68 searchNode(head, age\_key);

69 **break**;

70 }

**main.c**

**Output:**



**Post Lab:**

**Delete the last node of the linked list.**

**Program:** In this program, a function ***“deleteLastNode”*** takes first node of the linked list as its argument. Function ***“isEmpty”*** checks if the list is empty otherwise pointers *‘toDelete’* and *‘secondLastNode’* are initialized as *‘first’*. While loop then iterates till **‘next’** pointer of *‘toDelete’* doesn’t point to **NULL** i.e. until second last node is found. If the node to be deleted is the first node then *‘t*o*Delete’* is equal to *‘first’* thus *‘*first’ is made to point **NULL** otherwise *‘secondLastNode’* is made to point **NULL** and memory location of *‘toDelete’* is freed using **‘free( )’**. Finally, the message of successful deletion is printed on the console.

20 **void** deleteLastNode(**struct** employee \* first);

**employee.h**

105 **void** deleteLastNode(**struct** employee \* first)

106 {

107 **struct** employee \*toDelete, \*secondLastNode;

108

109 //Checks whether list is empty

110 **if**(isEmpty(first))

111 {

112 printf("List is empty \n");

113 }

114 **else**

115 {

116 toDelete = first;

117 secondLastNode = first;

118

119 /\* Traverse to the last node of the list \*/

120 **while**(toDelete->next != NULL)

121 {

122 secondLastNode = toDelete;

123 toDelete = toDelete->next;

124 }

125

126 **if**(toDelete == first)

127 {

128 first = NULL;

129 }

130 **else**

131 {

132 /\* Disconnect link of second last node with last node \*/

133 secondLastNode->next = NULL;

134 }

135

136 /\* Delete the last node \*/

137 free(toDelete);

138

139 }

140 }

**employee.c**

57 **case** 4: **/// Delete the last item from the list**

58 {

59 deleteLastNode(head);

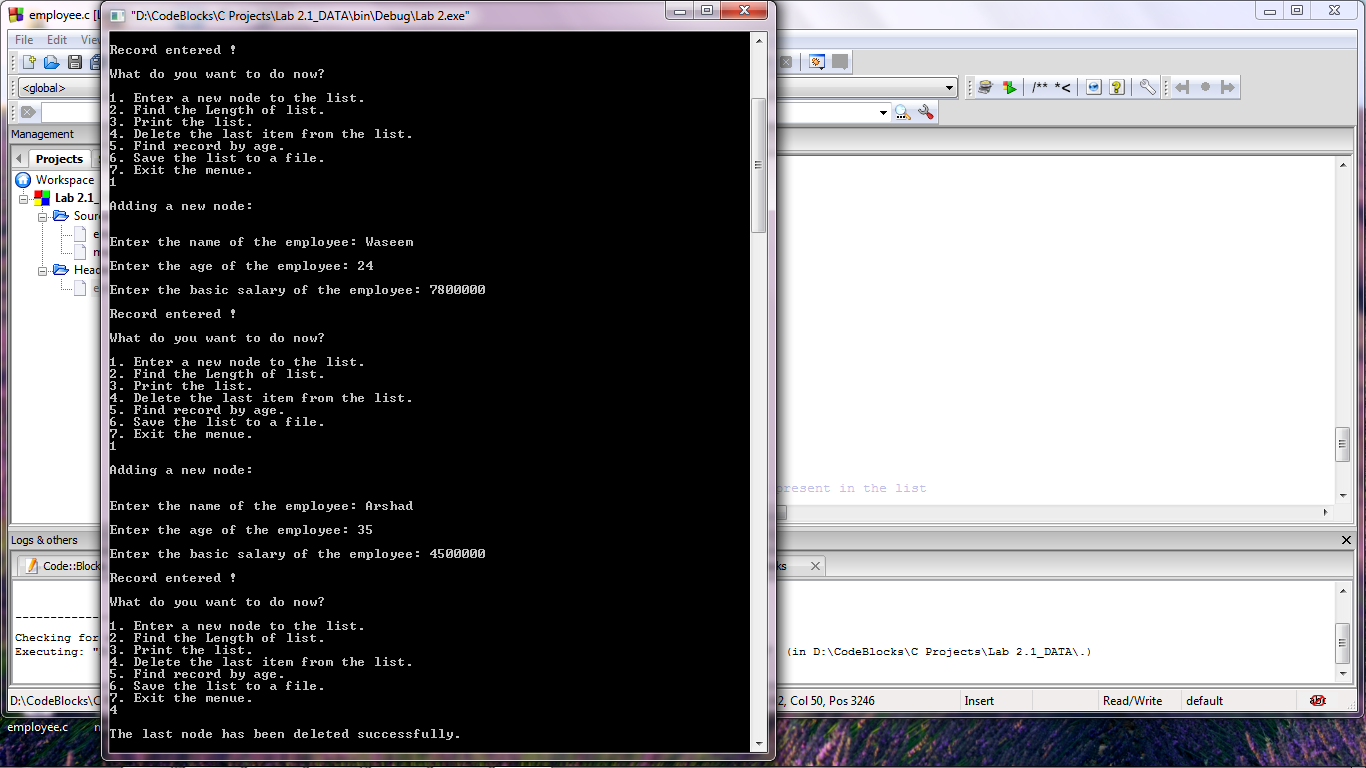
60 printf("\nThe last node has been deleted successfully.\n");

61 **break**;

62 }

**main.c**

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE END**