National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B.Tech.(CSE)
CS2092D Programming Laboratory
Assignment 5B

Submission deadline (on or before):

- 24.09.2023, 11:00 PM

Policies for Submission and Evaluation:

- You must submit your assignment in the Eduserver course page, on or before the submission deadline.
- Ensure that your programs will compile and execute without errors using gcc compiler.
- During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
- Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

Naming Conventions for Submission

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip`

  (Example: *ASSG5B_BxxyyyyCS_LAXMAN.zip*). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.
- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: *ASSG5B_BxxyyyyCS_LAXMAN_1.c*). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity_new.pdf.

General Instructions

- Programs should be written in C language and compiled using gcc compiler. **Submit the solutions to the questions through the submission link in Eduserver**.
- Check your programs with sufficiently large values of inputs with in the range as specified in the question.
- Global and/or static variables should not be used in your program.

QUESTIONS

1. A SINGLY LINKED LIST L is a data structure in which the objects are arranged in a linear order. Each node of a SINGLY LINKED LIST L is an object with an attribute *key* and one pointer attribute, *next*. Given a node x in the list, $x.next$ points to its successor in the linked list. An attribute $L.head$ points to the first node of the list.

   Write a menu driven program to implement an unsorted SINGLY LINKED LIST L. Your program must contain the following functions:

   (In the function prototypes, L, k, x and y denote a Singly Linked List, an integer and nodes of L respectively. All operations should be done in a single pass and all keys are distinct).

   - MAIN() - repeatedly reads a character 'f', 't', 'a', 'b', 'd', 'i', 'l', 's' or 'e' from the terminal and calls the sub-functions appropriately until character 'e' is entered.
   - CREATE-NODE(k) - creates a new node with *key* k and pointer *next* of node points to NULL. This procedure returns a pointer to the new node.
   - LIST-INSERT-FRONT(L, x) - inserts x to the front of L.
   - LIST-INSERT-TAIL(L, x) - inserts x as the last node of L.
   - LIST-INSERT-AFTER(L, x, y) - inserts the node x after the node y in L.
     Hint: First invoke LIST-SEARCH() function to locate the node y.
   - LIST-INSERT-BEFORE(L, x, y) - inserts the node x before the node y in L.
     Hint: Insertion of a node x before a node y can be done by locally storing a pointer to the current node before moving to the next node.
   - LIST-DELETE(L, x) - deletes the node x from L.
     Hint: First invoke LIST-SEARCH() function to locate the node x.
   - LIST-DELETE-FIRST(L) - deletes the first node from L.
   - LIST-DELETE-LAST(L) - deletes the last node from L.
   - LIST-SEARCH(L, k) - searches for a node with key k in L by doing a simple linear search, and if found, returns a pointer to this node. If a node with key k is not present in the list, or if the list is empty, the procedure returns NIL.
   - LIST-DISPLAY(L) - prints the list L
   - LIST-REVERSE(L) - reverses the list and prints it.
   - LIST-REVERSE-EVEN(L) - reverses the order of the elements in the even positions of the list and prints it (starting the index with one).

   Note:- For every INSERT operation, the node x is created by calling CREATE-NODE() function.

   Input format:

   - Each line contains a character from 'f', 't', 'a', 'b', 'd', 'i', 'l', 's', 'r', 'ds', 're' or 'e' followed by zero, one or two integers. The integers, if given, are in the range $[-10^6, 10^6]$.
   - Character 'f' is followed by an integer separated by space. In this operation, the node with this integer as key is inserted to the front of L.
   - Character 't' is followed by an integer separated by space. In this operation, the node with this integer as key is inserted to the tail of L.
   - Character 'a' is followed by two integers separated by space. In this operation, the node with the first integer as key is inserted after the node with second integer as key.
   - Character 'b' is followed by two integers separated by space. In this operation, the node with the first integer as key is inserted before the node with second integer as key.
   - Character 'd' is followed by an integer separated by space. In this operation, the node with this integer as key is deleted from L and the deleted node's key is printed.

- Character 'i' is to delete the first node from L and print the deleted node's key.
- Character 'l' is to delete the last node from L and print the deleted node's key.
- Character *'s'* is followed by an integer separated by space. This operation is to find the node with this integer as key in L.
- Character *'r'* This operation is to reverse the list L and to print it.
- Character *'ds'* This operation is to print the list L.
- Character *'re'* This operation is to reverse the elements in the even positions of the list L and to print the new list, starting the index with one.
- Character 'e' is to 'exit' from the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.
- For options 'd', 'i' and 'l' , print the deleted node's key. If a node with the input key is not present in L or L is empty, then print -1.
- For option 's', if the key is present in L, then print 1. If key is not present in L or L is empty, then print -1.

Sample Input :

```
f 7
t 10
a 11 7
b 12 11
d 10
i
l
s 12
s 6
t 15
f 14
f 20
ds
r
re
e
```

Sample Output:

```
10
7
11
1
-1
20 14 12 15
15 12 14 20
15 20 14 12
```

2. A Doubly linked list L is a data structure in which the objects are arranged in a linear order. Each node of a Doubly linked list L is an object with an attribute *key* and two other pointer attributes: *next* and *prev*. Given a node x in the list, *x.next* points to its successor in the linked list, and *x.prev* points to its predecessor. An attribute *L.head* points to the first node of the list.

   Write a menu driven program to implement an unsorted Doubly Linked List L. Your program must contain the following functions: (In function prototypes, L, k, x and y denote a Doubly

Linked list, an integer and the nodes of L respectively. All operations should be done in a single pass and all keys are distinct).

- MAIN() - repeatedly reads a character 'f', 't', 'a', 'b', 'd', 'i', 'l', 's' , 'rn' ,'ds' or 'e' from terminal and calls the sub-functions appropriately until character 'e' is entered.
- CREATE-NODE(k)- Creates a new node with *key* k and the pointers *next* and *prev* of node points to NULL. This procedure returns a pointer to the new node.
- LIST-INSERT-FRONT(L, x) - inserts node x to the front of L.
- LIST-INSERT-TAIL(L, x) - inserts x as the last node of L.
- LIST-INSERT-AFTER(L, x, y) - inserts node x after node y in L.
  Hint: First invoke LIST-SEARCH() function to locate the node y.
- LIST-INSERT-BEFORE(L, x, y) - inserts node x before node y in L.
  Hint: Insertion of a node x before a node y can be done by locally storing a pointer to the current node before moving to the next node.
- LIST-DELETE(L, x) - deletes node x from L.
  Hint: First invoke LIST-SEARCH() function to locate the node x.
- LIST-DELETE-INITIAL(L) - deletes the first node from L.
- LIST-DELETE-LAST(L) - deletes the last node from L.
- LIST-SEARCH(L, k) - searches for a node with key k in L by a simple linear search, and if found, returns a pointer to this node. If a node with key k is not present in the list, or the list is empty, then the procedure returns NIL.
- LIST-DISPLAY(L) - prints the list L
- LIST-REVERSE-NEGATIVE(L) - reverses the order of the negative elements in the list and prints the entire list.

Note:- For every INSERT operation, the node x is created by calling CREATE-NODE() function.

Input format:

- Each line contains a character from 'f', 't', 'a', 'b', 'd', 'i', 'l', 's' , 'r' or 'e' followed by zero, one or two integers. The integers, if given, are in the range $[-10^6, 10^6]$.
- Character 'f' is followed by an integer separated by space. In this operation, the node with this integer as key is inserted into the front of L.
- Character 't' is followed by an integer separated by space. In this operation, the node with this integer as key is inserted into the tail of L.
- Character 'a' is followed by two integers separated by space. In this operation, the node with the first integer as key is inserted after the node with second integer as key into L.
- Character 'b' is followed by two integers separated by space. In this operation, the node with the first integer as key is inserted before the node with second integer as key into L.
- Character 'd' is followed by an integer separated by space. In this operation, the node with this integer as key is deleted from L and the deleted node's key is printed.
- Character 'i' is to delete the first node from L and print the deleted node's key.
- Character 'l' is to delete the last node from L and print the deleted node's key.
- Character *'s'* is followed by an integer separated by space. This operation is to find the node with this integer as key in L.
- Character *'ds'* This operation is to print the list L.
- Character *'rn'* This operation is to reverse the order of the negative elements in the list and to print the entire list.
- Character *'e'* is to 'exit' from the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.
- For options 'd', 'i' and 'l', print the deleted node's key. If a node with the input key is not present in L or L is empty, then print -1.
- For option 's', if the key is present in L, then print 1. If key is not present in L or L is empty, then print -1.

Sample Input

```
f 20
t 38
a 22 20
b 35 22
d 38
i
l
s 35
s 40
f -30
f -20
ds
rn
e
```

Sample Output

```
38
20
22
1
-1
-20 -30 35
-30 -20 35
```