Tutorial & Laboratory

Programming & Data Structure: CS11001/19001

Section - 4/D

DO NOT POWER ON THE MACHINE

Department of Computer Science and Engineering I.I.T. Kharagpur

Spring Semester: 2013 - 2014 (**27.03.2014**)

Download

Download the file date270314.pdf from Programming & Data Structures ... of

http://cse.iitkgp.ac.in/~goutam

View the file using the command acroread & or xpdf &

Laboratory Test II: 10^{th} April, 2013

- Lab Test I + 1-D array, recursive function, string, 2-D array, structures, data types, lists, sorting, searching.
- Do not carry any book or note book.
- Time: Two hours.
- Marks: 35

The test will start after the tutorial.

Dynamic Set of Data

A dynamic set is a finite collection of data that may change over time. Examples are data related to students of IIT Kharagpur, employees of IIT Kharagpur, stock of commodities in a store etc.

We consider a simple dynamic set of integers.

Dynamic Set of Data

A few of the basic operations on a dynamic set are as follows:

- Initialisation of the set (to an empty set).
- Test for an empty set.
- Insertion of an element in the set.
- Searching for an element in the set.

Dynamic Set of Data

- Deletion of an element from the set.
- Successor of a data in the set.
- Predecessor of a data in the set.
- Printing elements of the set etc.

Successor and predecessor operations are meaningful if set elements are ordered.

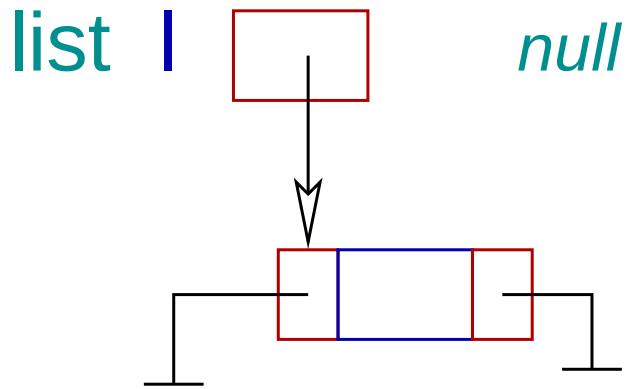
Dynamic Set of Integers

A dynamic set of integers may be implemented on a doubly-linked ordered list of self-referencing structures of the following type.

```
typedef struct node {
        int data;
        struct node *lP, *rP;
} node, *list;
```

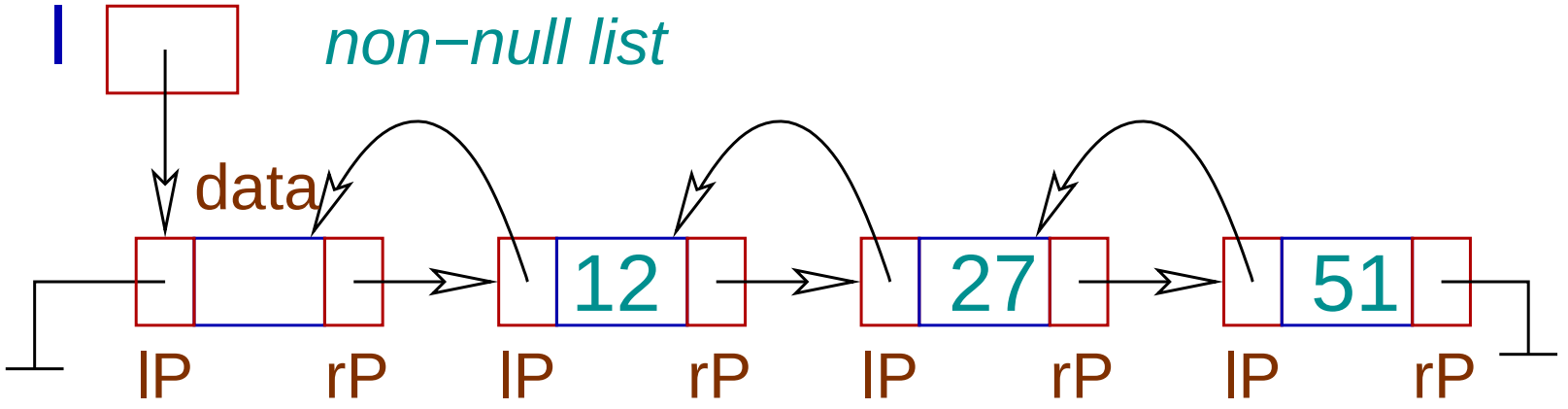
We use a dummy node as the header.

Doubly Linked List



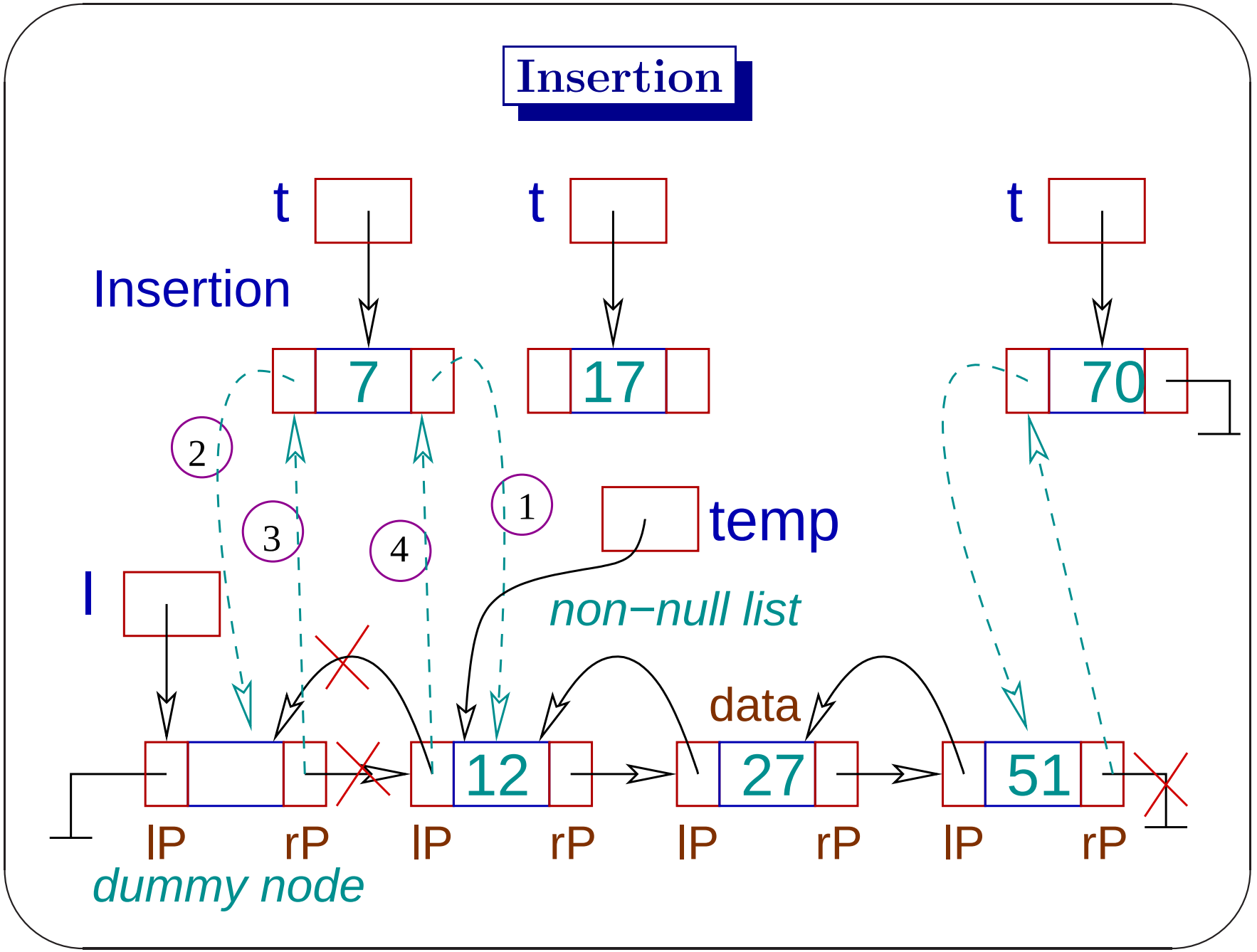
dummy node (Does not contain any data)

Ordered Doubly Linked List

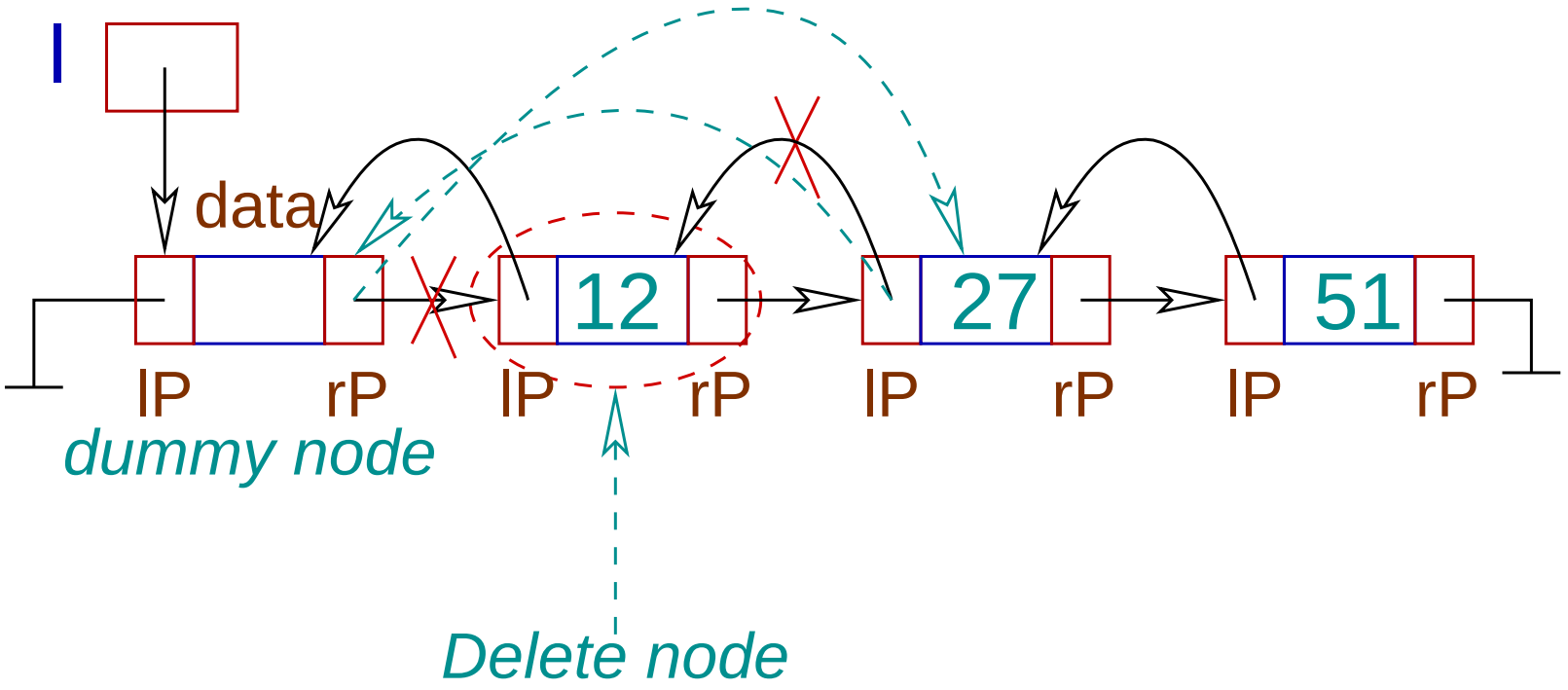


dummy node

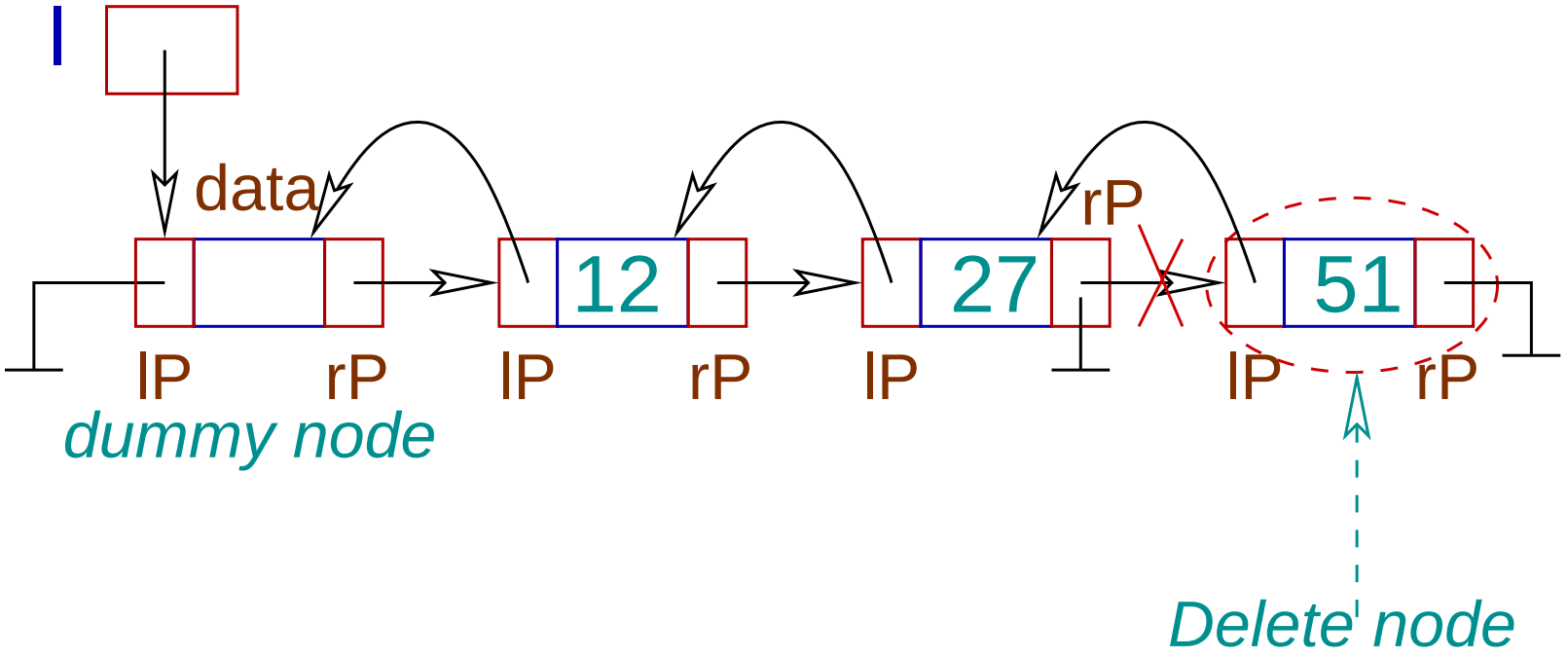
(Does not contain any data)


Insertion
t
t
t
Insertion
7
17
70
2
1
3
4
temp
l
non−null list
data
12
27
51
lP
rP
lP
rP
lP
rP
lP
rP
dummy node

Deletion



Deletion



Assignment XV

Write a C program to support the following function on a doubly-linked ordered list using dynamically created self-referencing structure shown earlier.
Two functions `int main()` and `list initList()` are supplied in a file `main15.c`.
Do not modify them. [Marks: $5+5+5+5+5+5+5$]

Task I

The function `int isEmptyList(list l)` will return 1 if the `list l` is empty, otherwise it returns 0. [Marks: 5]

Task II

The function `void printList(list l)` prints the elements of the `list l`. It ignores the content of the dummy node. [Marks: 5]

Task III

The function
`int insertList(list l, int data)`
dynamically created a new node, puts the `data` in it. The node is inserted in the `list l` maintaining increasing order, if the `data` is not already present in the list. It `returns 1` after a successful insertion, otherwise it `returns 0`.
[Marks: 5]

Task IV

The function
`list searchList(list l, int data)`
searches for the `data` in the `list l`. If the data is present in any node, it returns the address of the node. If the `data` is not present, it returns `NULL`. [Marks: 5]

Task V

The function
`int deleteList(list l, int data)` deletes the node containing `data` from the list. If the deletion is successful, it returns 1, otherwise it returns 0.
This function should use `searchList()` and free the node it has deleted. [Marks: 5]

Task VI

The function
`list successorList(list l, int d)` returns the address of the node where the data is larger than `d` and it is smallest among such data in the list. If no such node is not found, it returns `NULL`. [Marks: 5]

Task VII

The function
`list predecessorList(list l, int d)`
returns the address of the node where the data is smaller than `d` and it is largest among such data in the list. If no such node is not found, it returns `NULL`. [Marks: 5]

Note

In `main()` one can insert, delete, search, find successor, predecessor and also print data. The corresponding inputs are,

- I 30 - insert 30
- D 30 - delete 30
- S 30 - search for 30
- U 30 - successor of 30
- R 30 - predecessor of 30

- P - print the list
- E - exit

Test Data

i 27, i -2, i -7, i 10, p, d 27, i 27, p, d -7, i -7, p,
d 10, i 10, p, u -10, u -7, u -1, u 20, u 27, u 30,
$\cdots$ similarly for predecessor.

Submission by ftp

```
$ ftp 10.5.17.186
Connected to 10.5.17.186.
220---------- Welcome to Pure-FTPd ----
220-You are user number 1 of 50 allowed.
220-Local time is now 07:54. ....... 21.
220-IPv6 connections ........
220 ... disconnected .. inactivity.
Name (10.5.17.186:...): pds
```

Submission by ftp

```
331 User pds OK. Password required
Password: pds04
230-User pds has group access to:  pds
230 OK. Current restricted directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd assignment15
250 OK. Current directory is /assignment15
```

Submission by ftp

```
ftp> put D0615.c
local: D0615.c remote: D0615.c
200 PORT command successful
150 Connecting to port 47093
226-File successfully transferred
226 0.001 seconds .. 39.00 Kbytes ..
27 bytes sent in 0.00 secs (1098.6 kB/s)
ftp> bye
21-Goodbye. .....
221 Logout.
$
```