National Institute of Technology, Calicut

Department of Computer Science and Engineering

CS2094D – Data Structures Lab

Assignment-3 (Common Questions for Main and Advanced Batch)

Policies for Submission and Evaluation

You must submit your assignment in the moodle (Eduserver) course page, on or before the submission deadline. Also, ensure that your programs in the assignment must compile and execute without errors in Athena server. During evaluation your uploaded programs will be checked in Athena server only. Failure to execute programs in the assignment without compilation errors may lead to zero marks for that program. Your submission will also be tested for plagiarism, by automated tools. In case your code fails to pass the test, you will be straightaway awarded zero marks for this assignment and considered by the examiner for awarding F grade in the course. Detection of ANY malpractice regarding the lab course will also lead to awarding an F grade.

Naming Conventions for Submission

Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar or .tar.gz). The name of this file must be ASSG<NUMBER>_<ROLLNO>_<FIRSTNAME>.zip For example: ASSG4_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>_<FIRSTNAME>_<PROGRAM-NUMBER>.<extension> (For example: ASSG4_BxxyyyyCS_LAXMAN_1.c). If there is a part a and a part b or a particular question, then name the source files for each part separately as inASSG4_BxxyyyyCS_LAXMAN_1b.c.

If you do not conform to the above naming conventions, your submission might not be recognized by some automated tools, and hence will lead to a score of 0 for the submission. So, make sure that you follow the naming conventions.Standard of Conduct Violations 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.pdf .

Assignment Questions

1. Write a program that implements the disjoint-set data structure using rooted forests. Also,write functions to implement the ranked union and path compression heuristics on your data structure, and compute the efficiency of the disjoint set data structure find operation by applying neither, either or both of the heuristics, by counting the total number of data accesses performed over the course of the program.

Your program must support the following functions:

● **makeset(x)** creates a singleton set with element x .
● **find(x)** finds the representative of the set containing the element x
● **union(x,y)** merges the sets containing elements x and y into a single set. The representative of the resultant set is assigned with find(x) , unless the ranked union heuristic is used and the ranks of both find(x) and find(y) are different. Otherwise,the representative is assigned in accordance with the ranked union heuristic.

Note that looking up an element in the data structure must be done in O(1) time.

Input format
Your program should read input from **input.txt** and write the output to **output.txt**
The input consists of multiple lines, each one containing a character from {'m', 'f', 'u', 's'}followed by zero, one or two integers. The integer(s), if given, is in the range 0 to 10000.

● Call the function makeset(x) if the input line contains the character 'm' followed by an integer x . Print “PRESENT” if x is already present in some set, and the value of x , otherwise.

● Call the function find(x) if the input line contains the character 'f' followed by an integer x Output the value of find(x) if x is found, and “NOT FOUND”, otherwise.

● Call the function union(x,y) if the input line contains the character 'u' followed by space separated integers x and y . Print “ERROR”, without terminating, if either x or y isn’t present in the disjoint set. Print find(x) itself if find(x)=find(y) . Otherwise, print the representative of the resultant set. The representative of the resultant set is assigned with find(x), unless the ranked union heuristic is used and the ranks of both find(x) and find(y) are different. Otherwise, the representative is assigned in accordance with the ranked union heuristic.

● If the input line contains the character 's', print the number of data accesses performed by the find commands by each of the data structures over the course of the program and terminate.

Output format

The output consists of multiple lines of space-separated columns. The columns correspond

to the following disjoint-set data structures:
● a. with neither ranked union nor path compression applied.
● b. with only ranked union applied.
● c. with only path compression applied.
● d. with both ranked union and path compression applied.
Each line in the output contains the output of the corresponding line in the input, after applying to the respective data structures. The final line of the output contains the number of data accesses performed by the find commands by each of the data structures over the course of the program

Sample Input	**Sample Output**
m 1	1
m 2	2
m 3	3
m 4	4
m 5	5
m 6	6
m 7	7
m 8	8
m 9	9
u 1 2	1 1 1 1
u 3 4	3 3 3 3
u 5 6	5 5 5 5
u 7 8	7 7 7 7
u 9 8	9 7 9 7
u 6 8	5 5 5 5
u 4 8	3 5 3 5
u 2 8	1 5 1 5
f 9	1 5 1 5
m 10	10
u 10 9	10 5 10 5
s	38 32 33 30

2. Write a menu driven program to implement all the following operations on an AVL tree. Insertion, Deletion, Searching and Traversal. The input should be read from the file ***input.txt*** and output should be written to the file ***output.txt*.** Your program should include the following functions.

insert(A, element) – Inserts the data specified by element into tree A.

deleteNode(A,element) – Removes and returns the first-inserted element of AVL tree(A) if AVL tree is not empty, otherwise print 'EMPTY'.

getBalance(A,k) – prints the balance factor of the node k in the tree A. Balance factor will be an integer, which is calculated for node as **‘** height(left subtree) - height(right subtree) ‘

leftRotate(A,k) – Perform left rotation in the tree A, with respect tot node k

rightRotate(A,k) – Perform right rotation in the tree A, with respect to node k

isAVL(A) – checks if the tree pointed by A is an AVL tree or not.

printTree(A) – prints the tree given by A in the paranthesis format as **Tree t is represented as (t(left-subtree)(right-subtree)). Empty parentheses ( ) represents a null tree. There is no space in between paranthesis or numbers.Leaf node l is represented as (l)**

Search(A, element) - Returns TRUE if element is there in the tree A else returns FALSE

Input File Format

a) The input consists of multiple lines, each line of the input contains the value k first and then corresponding key for operations

Operations are:

If k=1 Insert the element into tree
If k =2 delete the node
if k =3 left rotate tree with respect to given node
if k =4 right rotate tree with respect to given node
if k = 5 print balance factor with respect to given node
if k = 6 print the tree as per the format specified above.
If k = 7 check if the tree is AVL, if yes print TRUE, else print FALSE
if k =8 search the given key in the tree, if present print TRUE, else print FALSE
if k = 9 terminate the execution of the program

Output File Format:

The output (if any) of each command should be printed on a separate line. S

S**ample Input:**

```
1 9
1 5
1 10
1 0
1 6
1 11
1 -1
1 1
1 2
```

```
6
8 10
8 100
7
5 11
9
```

S**ample Output:**

```
(9(1(0(-1)())(5(2)(6)))(10()(11)))
TRUE
FALSE
TRUE
0
```

3. Write a menu driven program to implement binomial heap .

Your Program should read input from **input.txt** and write output to **output.txt**

Your program must implement following fucntions.

1.**MAKE-HEAP()** creates and returns a new heap containing no elements.

2.**INSERT(H, x)** inserts node x into heap H.

3. **MINIMUM(H)** returns a pointer to the node in heap H whose key is minimum

4.**EXTRACT-MIN(H)** deletes the node from heap H whose key is minimum, returning a pointer to the node.

5.**UNION(H1, H2)** creates and returns a new heap that contains all the nodes of heaps H1 and H2. Heaps H1 and H2 are “destroyed” by this operation.

6.**DECREASE-KEY(H, x, k)** assigns to node x within heap H the new key value k, which is assumed to be no greater than its current key value.1

7.**DELETE(H, x)** deletes node x from heap H.

Input Format

The input consists of multiple lines, each one containing a character from {'i', 'p', ‘d’, 'm', 'x' , ‘s’}followed by zero, one integer.
i k inserts ‘k’ into the heap
d k – deletes node k from the heap
m prints the minimum value in the heap
p – prints the binomial heap
m – prints the minimum element in the binomial heap
x- extracts the minimum element from the heap
s- stops the execution

Sample Input

i 10

```
i 20
i 30
i 40
i 50
p
m
x
p
s
```

Sample Output

```
50 10 30 20 40
10
20 30 40 50
```

Note:- In print function , level order traversal is used.

4. Write a program that implements the disjoint-set data structure using linked list. Also,write functions to implement the weighted- union heuristic on your data structure.

Your program must support the following functions:

● **makeset(x)** creates a singleton set with element x .
● **find(x)** finds the representative of the set containing the element x
● **union(x,y)** merges the sets containing elements x and y into a single set. The representative of the resultant set is assigned with find(x)

Input format

Your program should read input from **input.txt** and write the output to **output.txt**
The input consists of multiple lines, each one containing a character from {'m', 'f', 'u', 's'}followed by zero, one or two integers. The integer(s), if given, is in the range 0 to 10000.

● Call the function makeset(x) if the input line contains the character 'm' followed by an integer x . Print “PRESENT” if x is already present in some set, and the value of x , otherwise.

● Call the function find(x) if the input line contains the character 'f' followed by an integer x Output the value of find(x) if x is found, and “NOT FOUND”, otherwise.

● Call the function union(x,y) if the input line contains the character 'u' followed by space separated integers x and y . Print “ERROR”, without terminating, if either x or y isn’t present in the disjoint set. Print find(x) itself if find(x)=find(y) . Otherwise, print the representative of the resultant set. The representative of the resultant set is assigned with find(x). For weighted union heuristic implementation,if both sets have the same number of elements, then the one with greater representative value is joined at the tail of the other set, and the lesser representative value becomes the resultant representative.

● If the input line contains the character 's', - terminate the program execution.

Output format

The output consists of multiple lines of space-separated columns. The columns correspond
to the following disjoint-set data structures:

- a. Withour weighted union heuristic
- b. with weighted union heuristic

Each line in the output contains the output of the corresponding line in the input, after applying to the respective data structures.

Sample Input	Sample Output
m 1	1
m 2	2
m 3	3
m 4	4
m 5	5
m 6	6
m 7	7
m 8	8
m 9	9
u 1 2	1 1
u 3 4	3 3
u 5 6	5 5
u 7 8	7 7
u 8 9	7 7
u 6 8	5 7
u 4 8	3 7
u 2 8	1 7
f 9	1 7
s	