IIT PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Palakkad
CS2130: Data Structures and Algorithms Lab
Lab 12 (Sorting)

2 Nov, 2018

Time: 3 hrs

General Instructions

- You are expected to follow good programming practices (refer to the book titled *C How to Program* by Dietel and Dietel). Your programs should handle all kinds of test cases - positive, negative, corner cases/boundary conditions, good input, bad input and are never expected to crash on any input.
- You are also expected to provided a high level description of the functions used in your programs. There is no restriction on the format except that it should be concise and precise. Further, the running time and additional space (excluding the space for the arguments) used by the functions should be clearly stated and justified.
- **Please note that poor programming practices and improper documentation can attract a penalty of up to 20% of the marks allocated to a question.**

1. (30 points) Write a program to implement a hash table to store distinct keys that are non-negative integers with collisions resolved using the linear probing method.

   Input-Output Format: The first line of the input contains the number **m** denoting the number of slots in the hash table. Each subsequent line contains a specific operation to be performed on the hash table. Assume that the slots of the hash table are numbered from **0** to **m-1**. Cells that are empty are assumed to contain **-1**.

   INS x. Insert **x** into the table using the probe sequence $<$ **h(x, 0), . . . , h(x, m-1)** $>$ where **h(x, i) = (h(x) mod m + i) mod m**. Print the sequence of probes made in the process.
   DEL x. Deletes **x** from the table by replacing the contents of the cell that contained **x** by a deletion marker denoted by **-2**. Print the sequence of probes made in the process.
   SRCH x. Searches for the key **x** in the table. Print the sequence of probes made in the process.

   In the output, each probe is denoted by **i(x)** denoting that the **i**th cell was probed and the key it contained was **x**.

   ```
   Sample Input 1:
   17
   INS 10
   INS 5
   INS 1
   INS 7
   INS 100
   ```

Question 1 continues on the next page. . .

```
INS 25
INS 10
INS 7
DEL 7
SRCH 23
SRCH 5
INS 15
INS 14
INS 90
INS 17
INS 50
INS 55
SRCH 17

Expected Output:
10(-1)
5(-1)
1(-1)
7(-1)
15(-1)
8(-1)
10(10)
7(7)
7(7)
6(-1)
5(5)
15(100) 16(-1)
14(-1)
5(5) 6(-1)
0(-1)
16(15) 0(17) 1(1) 2(-1)
4(-1)
0(17)

Sample Input 2:
10
INS 10
INS 5
INS 1
INS 7
INS 100
INS 25
INS 10
INS 7
DEL 7
SRCH 23
```

Continued on next page...

```
SRCH 5
INS 15
INS 14
INS 90
INS 17
INS 24
SRCH 17

Expected Output:
0(-1)
5(-1)
1(-1)
7(-1)
0(10) 1(1) 2(-1)
5(5) 6(-1)
0(10)
7(7)
7(7)
3(-1)
5(5)
5(5) 6(25) 7(-2)
4(-1)
0(10) 1(1) 2(100) 3(-1)
7(15) 8(-1)
4(14) 5(5) 6(25) 7(15) 8(17) 9(-1)
7(15) 8(17)
```

2. (40 points) Write a program to implement *quick sort* (in ascending order). Make sure that you always pick the last element at the pivot. The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller and equal elements before x, and put all greater elements after x. All this should be done in linear time. (40 points)

   Input format: The first line of the input contains n indicating the number of elements in the array. Each subsequent line contains an element of the array.

```
Sample Input
10
3
2
10
9
6
5
8
7
4
```

Continued on next page...

```
1
```

Output format: Print the partitioning index, in a new line, after every call to the partition function.

3. (30 points) Write a program to merge overlapping intervals in the given set of intervals. Full credit will be awarded only if your solution has a time complexity of $O(n \log n)$. (30 points)

Input format: The first line mentions the number of intervals n. The $(i+1)^{\text{th}}$ of the input contains 2 integers a_i and b_i ($a_i \leq b_i$) separated by a space denoting the interval $[a_i, b_i]$. The first integer denote the start of the interval and the second integer denotes its end.

```
Sample input
6
6 8
1 9
2 4
4 7
10 12
```

Output format: Print the collection of merged interval in increasing order of the start time. One interval per line. Each line should have two integers separated by a space. The first integer denote the start of the interval and the second integer denotes its end.

```
Sample output
1 9
10 12
```