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

Submission deadline (on or before):

- 07.10.2020, 10:00 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 in the Linux platform.
- 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: *ASSG*1_*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: *ASSG*1_*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 C compiler in Linux platform. **Submit the solutions to questions 1 and 2 through the submission link in Eduserver. You should complete the third program before the lab session. During the evaluation we may be asking you to modify any of these three programs**.

The Sorting Problem can be formally stated in terms of the input/output relationship as follows:

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$

Output: A permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

QUESTIONS

General Note: For the following programs, do not declare the array as a global variable. You may pass the arrays as function arguments by using the concept of pointers.

1. Write a program that uses the HEAP-SORT algorithm for sorting a given input sequence of integers present in an array A in non-decreasing order and prints the number of comparisons performed during sorting. Your program must contain the following functions: (the notation $A[i..j]$ denotes the sub-array of A, contained within the i^{th} and j^{th} indices, both inclusive).

   - A recursive function MAX-HEAPIFY(A, i) that takes as input an array A and lets the value at $A[i]$ "float down" in the max-heap so that the subtree rooted at index i obeys the max-heap property.
   - A function BUILD-MAX-HEAP(A) that takes as input an array A and build a max-heap on the input array $A[1...n]$ where n is equal to $A.length$.
   - A function HEAPSORT(A) that takes as input an array A and sorts an array A in place.

   Input format:

   - The first line of the input contains an integer $n \in [0, 10^5]$, the size of the array A.
   - The second line lists the n elements in A, as space-separated integers in the range $[-10^3, 10^3]$.

   Output Format:

   - The first line of the output contains the elements of A in sorted order, separated by space.
   - The second line of the output contains the number of comparisons performed during sorting.

   Note:

   The number of comparisons made by Heap-Sort is highly dependent on its implementation. As such, we will be considering the number of comparisons as per the algorithm given in CLRS.

   Sample Input:

   8
   98 67 56 45 43 23 20 12

   Sample Output:

   12 20 23 43 45 56 67 98
   24

2. Write a menu driven program to implement a MIN-PRIORITY QUEUE using HEAP. A priority queue maintains a set S of elements, each with an associated value called a *priority*. For simplicity, you may assume that the priority value itself represents the element. Your program must contain the following functions.

   - MAIN() - repeatedly reads a character 'i', 'e', 'm ', 'd' or 's' from the terminal and calls the sub-functions appropriately until character 's' is entered.
   - MIN-HEAPIFY(A, i)- A recursive function that takes as input an array A and lets the value at $A[i]$ "float down" in the min-heap so that the subtree rooted at index i obeys the min-heap property. Note that, the heap may contain duplicate values. Hence during comparisons, if the parent value is greater than its children value (whose values are equal) then, left child value is exchanged with parent value.

- MIN-HEAP-INSERT(Q, k) - A function that inserts a new element with priority value k into the priority queue Q.
- HEAP-EXTRACT-MIN(Q) - A function that takes as input a priority queue Q, removes and prints the element with the lowest priority. If the priority queue is empty, print -1.
- HEAP-MINIMUM(Q) - A function that takes as input a priority queue Q, prints the element with the lowest priority without actually removing it from the priority queue. If the priority queue is empty, print -1.
- HEAP-DECREASE-KEY(Q, i, k) - A function that takes as input a priority queue Q, and integers i and k. It changes the priority of element at index i to a new value k. Assume that k is lower (in the sense of priority) than the current priority value present in the index i. The value i is within the range $[0, n-1]$, where n is the current size of the queue. Note that, after this operation the heap may contain duplicate values. Hence during comparisons, if a node's value is equal to its parent value then, no swap is permitted.

Input format:

- First line contains a character from 'i', 'e', 'm', 'd', 's' followed by at most two integers. The integers, if given, are in the range $[1, 10^6]$.
- Character 'i' is to insert the next integer, from the input into the priority queue. Character 'i' and integer are separated by space.
- Character 'e' is to remove and print the lowest key from the priority queue Q.
- Character 'm' is to print the lowest key from the priority queue Q without actually removing it.
- Character 'd' is to decrease the priority of an element in the priority queue. In this case, character 'd' is followed by two more integers, each separated by space. After this operation, the value at the position of first integer is decreased to the second integer.
- The character 's' is to 'stop' the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.
- For option 'e', remove and print the lowest key from the priority queue. If the priority queue is empty, then print -1.
- For option 'm', print the lowest key in the priority queue. If the priority queue is empty, then print -1.

Note:

In CLRS, MAX-PRIORITY QUEUE is discussed. CLRS also give an idea about how to implement MIN-PRIORITY QUEUE.

Sample Input :

```
i 100
i 110
i 95
e
e
i 120
e
i 30
m
d 0 25
m
e
```

e
e
s

Sample Output:

95
100
110
30
25
25
120
-1

3. A hospital emergency room treats patients according to the severity of his/her health problems. Whenever a new patient is admitted to the hospital, a *token number* is assigned to the patient based on the seriousness of illness. This token number indicates the priority value - lesser the value, higher the priority.

   Write a menu driven program to implement the above scenario as a *Patient Scheduling Application*, using a Priority Queue, Q (implemented as heap). Implement the following operations:

   - MAIN() - repeatedly reads a character 'i', 'e', 'm ', 'c', 'd' or 's' from the terminal and calls the sub-functions appropriately until character *'s'* is entered.
   - INSERT-PATIENT($Q,\ k$) - Inserts a new patient with token number k into the *Queue* Q.
   - EXTRACT-NEXT-PATIENT(Q) - Get the token number of the patient to be treated next and remove his/her priority from the queue.
   - GET-NEXT-PATIENT(Q) - Get the token number of the patient to be treated next, from the *Queue* Q without removing it.
   - CHANGE-TOKEN-NUMBER($Q,\ k,\ x$) - Change the token number of a patient from old value k to new value x in *Queue* Q. Assume that token number k is present in the *Queue* and x is lower than the current token number k.
   - DISPLAY-QUEUE(Q) - Display all patients' token numbers in the *Queue* Q in the order of their treatment (patient with lowest token number is treated first) without changing the queue.

   Input format:

   - The first line of the input contains a character from 'i', 'e', 'm', 'c', 'd', 's' followed by at most two integers. The integers, if given, are in the range $[1, 10^6]$.
   - Character 'i' is to insert the next integer, from the input into the *Queue*. Character 'i' and integer are separated by space.
   - Character 'e' is to remove and print the patient with lowest token number from the *Queue* Q.
   - Character 'm' is to print the patient with lowest token number from the *Queue* Q without removing it.
   - Character 'c' is to change the token number of a patient in the *Queue*. In this case, character 'c' is followed by two more integers, each separated by space. After this operation, the patient's token number with first integer is decreased to the second integer.
   - The character 'd' is to display all patients' token numbers in the *Queue* in the order of their treatment (Patient with lowest token number is treated first) without changing the queue.
   - The character 's' is to 'stop' the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.
- For option '*e*', remove and print the patient with lowest token number from the priority queue. If the *Queue* is empty, then print -1.
- For option '*m*', print the patient with lowest token number in the *Queue*. If the *Queue* is empty, then print -1.
- For option '*d*', display all patients' token numbers in the *Queue* in the order in which they will get treatment (patient with lowest token number is treated first). Each token number should be separated by space.

Sample Input :

```
i 2
i 15
i 5
e
i 4
m
c 5 1
d
e
e
e
e
s
```

Sample Output:

```
2
4
1 4 15
1
4
15
-1
```