National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B.Tech.(CSE)
CS2092D Programming Laboratory
Assignment #4

Submission deadline (on or before):

- 06.10.2021, 9:00 AM

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: *ASSG1_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: *ASSG1_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. 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 $n \in [1, 10^5]$, 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. Radhika works in the HR department of a software company situated in Bangalore. Her company came to NITC for the recruitment process. They conducted a written test for the students and published a rank list based on the test and other criteria for conducting the interview. There is no more than one student with the same rank, and every student is assigned a unique rank and ID. On the interview day, the students are requested to be in a queue according to their rank. When a new student arrives, he/she will be positioned in the queue based on his/her rank (smaller the value, higher the rank). Every time, the highest-ranked student from the queue will be called for the interview.

   As a placement cell coordinator, your job is to help Radhika to :

   - Place the arriving students in the priority queue accordingly. The highest-ranked student will be in the front (smaller the rank value, higher the priority).
   - Pick the highest-ranked student from the queue and send him to the interview panel. Then rearrange the queue so that the highest-ranked student comes to the front.

   Create an array of structures **S** that stores the following details of each student:

   Student_id
   Student_name
   Rank

   Write a menu driven program that implements priority queue (using heap) and the following functions as per the given function prototypes:

- **main()** - repeatedly reads a character **'e', 'i', 'l',** or **'t'** from the terminal and call the corresponding functions, as described in Input Format, until character 't' is entered.
- **ENTER(S)**- Add a student record to the Priority Queue S.
- **INTERVIEW(S)**- Find the highest-ranked student for the interview and display his/her Student ID. Rearrange the priority queue by removing the interviewed student's details.
- **LIST(S):** Display the details of the students yet to be called for interview in the order of their priority.

Input Format

The input consists of multiple lines. Each line starts with a character from {e, i, l, t} followed by zero or more integers or characters.

- Character **'e'** will be followed by an integer $\in [1, 10^4]$, specifying the *Student_id* of the student, a string of maximum 20 characters $\in [A - Z]$, specifying the *Student_name* of the student and an integer $\in [1, 10^4]$, specifying *Rank* of the student as space separated. Read and store the details of the student using the *ENTER()* function.
- Character **'i'** is to interview the top priority student from the queue and print the student ID using the *INTERVIEW()* function. If the queue is empty, print -1.
- Character **'l'** is to display the ID of all the remaining students in their order of priority using the *LIST()* function. If the queue is empty, then print -1.
- Character **'t'** to terminate the program.

Output Format

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

- For option **'i'**, print the ID and name of the student interviewed.
- For option **'l'**, print the ID of all students yet to be interviewed in their order of priority on the same line separated by a space.

Sample Input 1:

```
e 50 JOHN 7
e 24 MARK 15
e 35 SHON 9
i
i
e 36 SAMIHA 1
e 39 MIHIKA 12
i
e 42 ASHLEY 5
e 46 MARIA 20
e 49 ANIK 10
l
i
i
i
i
i
i
l
t
```

Sample Output 1:

```
50 JOHN
```

```
35 SHON
36 SAMIHA
42 49 39 24 46
42 ASHLEY
49 ANIK
39 MIHIKA
24 MARK
46 MARIA
-1
-1
```

3. Consider a set of Amoebas each having a unique name. Amoebas have a special property that two of them can combine together to form a bigger one. Write a program to combine the given set of amoebas until you are left with a single amoeba. The amoebas in the given set are combined based on the following criteria:

   - Amoeba with the smallest size will have the highest priority.
   - If two or more amoebas have the same size then amoeba who come first in the the alphabetical order of their names will have the highest priority.
   - At a time, only two of the amoebas who have the first two highest priorities can combine together to form the new amoeba.
   - The new amoeba can take the name of the amoeba who has the highest priority among the combining two amoebas.

   Create an array of structures W that stores the following details of each amoeba:

   Amoeba_Name
   Amoeba_Size

   Write a program that implements priority queue (using heap) and the following functions as per the given function prototypes:

   - **READ(a)** - Read the *Amoeba_Name* and *Amoeba_Size* of each amoeba to the structure **a**.
   - **INSERT(W, a)** - Insert amoeba a in to the priority queue W.
   - **COMBINE(W)** - Find the two amoebas with the highest priorities. Print their names in the decreasing order of their priority followed by the size of the newly formed amoeba. Return the details of the newly formed amoeba.

   Input format:

   - The first line of the input contains an integer $n \in [1, 10^3]$,the size of the structure array W.
   - The next n lines lists the elements in W where each line contains a string $S \in [A-Z, a-z]$, where $|S| \leq 20$ specifying the name of the amoeba *Amoeba_Name* followed by an integer $k \in [1, 10^3]$, specifying the size of an amoeba, *Amoeba_Size.*

   Output Format:

   - The $n-1$ lines of the output contains the names of the combining amoebas in the decreasing order of their priority followed by the size of the new amoeba.
   - The n^{th} line of the output contains the *Amoeba_Name* and *Amoeba_Size* of the final amoeba.

   Sample Input 1:

```
6
Akk 22
```

```
Crt 9
DDP 17
Mnl 39
HjG 4
Wss 4
```

Sample Output 1:

```
HjG Wss 8
HjG Crt 17
DDP HjG 34
Akk DDP 56
Mnl Akk 95
Mnl 95
```

Sample Input 2:

```
4
Sac 5
TaX 8
roM 20
PoP 10
```

Sample Output 2:

```
Sac TaX 13
PoP Sac 23
roM PoP 43
roM 43
```

4. 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
```