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):

- 12.09.2022, 11:55 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 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

- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: $ASSG1\_BxxyyyyCS\_LAXMAN\_1.c$).

  Create separate files for each question and place all of them in a single folder named

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>`

  (Example: $ASSG1\_BxxyyyyCS\_LAXMAN$). Then Zip the **folder**.

- Submit the single ZIP (.zip) file you have created(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.

  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.
- You shouldn't change the function prototype specified in the question.
- **Consider array indexes start at 0, unless otherwise specified.**

QUESTIONS

1. A class teacher wants to seat her students who have passed the Examination in the first row of the auditorium for a valedictory function. For that, she decided that the students having marks lesser than the class representative (CR) would sit right of the CR, and students with higher marks would sit left of the CR.
   Write a program that generates the required seating arrangement using the idea of **Quick Sort**.

   Your program should contain an *array of structures* **S** that stores the following details of each student:

   Student_Name

   Mark

   Input format:
   - The first line of the input contains an integer $n \in [1, 10^3]$, number of students in the class (the size of the structure array **S**).
   - The next n lines lists the elements in **S** where each line contains a string $X \in [A - Z]$ where $|X| < 30$ specifying the first name of the student $Student\_Name$ followed by an integer $k \in [210, 600]$, specifying the marks obtained by each student.
   - The last line contains the first name of the Class Representative, $C \in [A - Z]$ where $|C| < 30$.

   Output format:
   - Output contains the names of students in the order specified by the teacher, separated by single space.

   Sample Test Case #1
   Input:

```
8
JOHN 270
MARK 250
MIHIKA 312
SHANE 290
SAMI 210
ASHLEY 500
MARIA 420
ANIK 510
SHANE
```

   Output:

```
JOHN MARK SAMI SHANE MIHIKA ASHLEY MARIA ANIK
```

   Sample Test Case #2
   Input:

```
6
ASH 400
MAYA 420
MIHIKA 300
ANIK 500
SHANE 220
SAMI 210
MIHIKA
```

Output:
```
SHANE SAMI MIHIKA ANIK ASH MAYA
```

2. 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 $n \in [1, 10^3]$, 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
   ```

3. A batch of students is entering the playground. The teacher tries to group the students into teams as and when they arrive. If we have n students, the teacher will group first k students in Group 1, following k students in Group 2, and so on (n is always a multiple of k). The tallest student in each group becomes the captain. Assuming that each kid has a distinct height, your task is to determine the captain's height for each group using the idea of *Heap Sort* by reusing the code written for Question 2.

   For example, If n=6 and k=3, the heights of the students entering is $A = \{99, 89, 75, 106, 78, 67\}$ centimeters. Then we have two groups with heights - $\{99, 89, 75\}$ and $\{106, 78, 67\}$. Then the kid with a height of 99 cm becomes *Group 1* captain, and the kid with a height of 106 cm is the *Group 2* captain.

   Input format:
   - The first line of the input contains two integers $n \in [1, 10^3]$, the total number of students, and $k \in [1, n]$, the number of students in each group.
   - The second line lists the heights of n students in an array A as space-separated floating point numbers in two decimal points in the range $[50, 150]$.

   Output format:
   - Print (n/k) space-separated floating point numbers in two decimal places indicating the heights of each group captain.

   Sample Input:
   ```
   10 5
   100.25 100.75 98.50 76.78 105.45 103.23 89.79 67.78 106.25 103.75
   ```

Sample Output:
```
105.45 106.25
```

4. Given an integer array A of size n. Write a C program using Priority Queue (implemented as a heap) to returns the top k most frequent elements in the array A.
For example, If given $n = 8$, $A = \{11, 22, 33, 44, 44, 44, 33, 22\}$ and $k = 3$, you have to print the k most frequent elements, i.e., $\{44, 22, 33\}$

Input format:
- The first line of the input contains an integer $n \in [1, 10^3]$, the total number of elements in the array.
- The second line lists the n elements in an array A as space-separated integers in the range $[-1000, 1000]$
- The third line contains an integer $k \in [1, x]$ representing the number of top most frequent elements, where x is the number of distinct elements in the array A.

Output format:
- Print k space-separated integers on a single line which indicates the most frequent elements, in the order of their first appearance in the input array A if there is a tie in the frequency.

Sample Test Case #1
Input:
```
6
1 1 1 2 2 3
2
```

Output:
```
1 2
```

Sample Test Case #2
Input:
```
1
-99
1
```

Output:
```
-99
```

5. CPU Scheduling is the procedure that the CPU employs to decide the order in which processes should be executed when multiple processes are waiting for their turn to complete execution. Consider a CPU Scheduling Algorithm where the scheduling is based on the priority value assigned to each process. Now write a menu-driven program to implement a Priority-Based Scheduling Algorithm, using a Priority Queue, Q (implemented as a heap). Consider maximum possible size of the queue as 1000. Your program should include the following functions:

   - 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_PROCESS (Q, k) - Inserts a new process with priority k into Q.
   - EXTRACT_NEXT_PROCESS (Q) - Return the priority of the process to be scheduled next and remove it from Q.
   - GET_NEXT_PROCESS (Q) - Return the priority of the process to be scheduled next, from Q without removing it.
   - CHANGE_PRIORITY (Q, k, x) - Change a process's priority from old value k to new value x in Q.
   - DISPLAY_QUEUE(Q) - Display all priority of processes in Q in their scheduling order without changing Q.

   Note: Define the current number of elements in Q as a global variable.

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 $[0, 10^6]$.
- Character 'i' is to insert an integer - which is the priority value of the next process - into Q. Character 'i' and integer are separated by space.
- Character 'e' is to remove and print the process with lowest priority from Q.
- Character 'm' is to print the process with lowest priority from Q without Removing it.
- Character 'c' is to change the priority of a process in Q. In this case, character 'c' is followed by two more integers, each separated by space. After this operation, the process's priority is changed from the first integer to the second integer, thereby further altering the Q.
- The character 'd' is to display all process's priority from Q 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 process with lowest priority from Q. If Q is empty, then print -1.
- For option 'm', print the process with lowest priority without removing it from Q. If Q is empty, then print -1.
- For option 'd', display all process's priority in Q. Each priority should be separated by space. If Q is empty, then print -1.

Sample Input:

```
i 3
i 7
i 9
i 1
e
i 11
m
c 7 0
d
e
e
i 1
e
e
e
e
s
```

Sample Output:

```
1
3
0 3 9 11
0
3
1
9
11
-1
```