National Institute of Technology Calicut
Department of Computer Science and Engineering
B. Tech. (CSE) – Third Semester
CS2092D: Programming Laboratory
Assignment –2 PART–A

Submission deadline (on or before):
28th August 2019, 10:00:00 PM

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>_<FIRST-NAME>.zip (For example: ASSG2A_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>.<extension>

(For example: ASSG2A_BxxyyyyCS_LAXMAN_01.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 an F grade in the course. The department policy on academic integrity can be found at:

http://minerva.nitc.ac.in/cse/sites/default/files/attachments/news/Academic-Integrity_new.pdf

General Instructions

Programs should be written in C language and compiled using the C compiler in Linux platform. Invalid input should be detected and suitable error messages should be generated. Sample inputs are just indicative. Please do the programs in your free time either from System Software Lab (SSL) / Network Systems Lab (NSL) / Hardware & Embedded Systems Lab (HL), when the lab is not used for regular lab hours or do the programs using your own computer. Even if the programs work in your own computer, there is a chance that they may not work properly in the computers in SSL / NSL/ HL, due to some compatibility issues of the C compiler or the machine. Hence, before the evaluation day, check that your programs are ready for execution in the computers in NSL/SSL/HL.

Evaluation of questions from the following questions will be conducted on **29 th August 2019** Thursday 2.00 pm - 5.00 pm.

NOTE

For all the questions in this assignment, input should be read from a file and output should be written to a file.

Questions

1. Write a menu-driven C program to search for a given integer ***k***, in an array of integers ***A*** of size ***n***. Print the index (starting from 0) of the first occurence of ***k***, if ***k*** is present in ***A*** else print '-1'. Assume that $1 \leq n \leq 100$.

Your program should implement the following functions:

main() - repeatedly reads a character ***'r'***, ***'s'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.

read(n) – read ***n*** integers from the terminal and store it in an array ***A***.

search(A, k) – search the integer ***k*** in array ***A***, and return the index of first occurrence of ***k,*** if ***k*** is present in ***A***, otherwise return -1.

display(A) – display the elements of ***A***.

INPUT FORMAT:

First line contains a character from {***'r', 's', 'd' ,'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of space separated ***n*** integers, to be read into the array.

• Character ***'s'*** is followed by an integer ***k***, the value to be searched in the array.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT:

For option ***'s'***, print the index of first occurrence of ***k***, if ***k*** is present in ***A***, and -1 if ***A*** does not contain ***k***.

For option ***'d'***, print the elements of array separated by space in a new line.

Sample Input:

```
r 7
12 35 59 59 60 73 90
s 59
d
s 100
t
```

Sample Output:

```
2
12 35 59 59 60 73 90
-1
```

2. Consider an array of integers sorted in ascending order and has been circularly shifted from left to right a number of times. Write a C program to find the number of times the array has been shifted using **binary search**.

Your program should implement the following functions.

main() - repeatedly reads a character ***'r', 's'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.
read(n) – read ***n*** integers from the terminal and store it in an array ***A***.
searchShifts(A, m), where ***m*** is the size of array ***A,*** and the function finds the number of times the array has been shifted and returns the value.
display(A) – display the elements of A.

INPUT FORMAT :

First line contains a character from {***'r', 's', 'd' ,'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of space separated ***n*** integers, to be read into the array.

• Character ***'s'*** is to find the number of times the given array has been shifted, and displays the result.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT :

For option ***'s'***, print the number of times the array has been shifted.

For option ***'d'***, print the elements of array separated by space in a new line.

Sample Input:

```
r 6
15 18 2 3 6 12
d
s
t
```

Sample Output:

```
15 18 2 3 6 12
2
```

3. An array is bitonic if it is comprised of an increasing sequence of integers followed immediately by a decreasing sequence of integers. Given a bitonic array ***A*** of ***n*** distinct integers, determine whether the given integer ***k*** is in the array. Assume that the array index starts from 0 and $1 \leq n \leq 100$. Implement this algorithm using **binary search.**

Your menu-driven program should implement the following functions:

main() - repeatedly reads a character ***'r', 's'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.

read(n) – read ***n*** integers from the terminal and store it in an array ***A***.

search(A, k) – search the integer ***k*** in array A. The function returns the index of ***k*** if ***k*** is present in ***A***, otherwise return -1.

display(A) – display the elements of A.

INPUT FORMAT:

First line contains a character from {***'r', 's', 'd' ,'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of ***n*** integers separated by a space, to be read into the array.

• Character ***'s'*** is followed by an integer ***k***, the value to be searched in the array.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT:

For option ***'s'***, print the index of first occurrence of ***k***, if ***k*** is present in ***A***, and -1 if ***A*** does not contain ***k***.

For option ***'d'***, print the elements of array separated by space in a new line.

Sample Input:

```
r 7
12 35 59 90 73 60 50
s 73
d
s 100
t
```

Sample Output:

```
4
12 35 59 90 73 60 50
-1
```

4. Given an array ***A*** of ***n*** integers, sort the array in descending order using insertion sort. Print the number of comparisons, number of shifts (any change in position of an element is considered as a shift), and the elements of the array in each iteration. Assume that $1 \leq n \leq 100$.

Your program should implement the following functions:

main() - repeatedly reads a character ***'r'***, ***'s'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.
read(n) – read ***n*** integers from the terminal and store it in an array ***A***.

sort(A) – perform insertion sort on array ***A***, and print the number of comparisons and number of shifts in each iteration. Use display() function to print the array contents at the end of each iteration.

display(A) – display the elements of ***A***.

INPUT FORMAT:

First line contains a character from {***'r'***, ***'s'***, ***'d'*** ,***'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of ***n*** integers separated by a space, to be read into the array.

• Character ***'s'*** is to sort the given array using insertion sort.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT:

For option ***'s'***, print the output for insertion sort at the end of each iteration. Display the number of comparisons and the number of shifts in the same line (space separated) followed by the elements of the array in the next line.

For option ***'d'***, print the elements of array separated by space in a new line.

Sample Input:

```
r 7
12 35 59 60 7 90 43
d
s
t
```

Sample Output:

```
12 35 59 60 7 90 43
1 2
35 12 59 60 7 90 43
2 3
59 35 12 60 7 90 43
3 4
60 59 35 12 7 90 43
1 0
60 59 35 12 7 90 43
5 6
90 60 59 35 12 7 43
4 4
90 60 59 43 35 12 7
```

5. You are given two sorted arrays of integers, ***X*** and ***Y*** of size ***m*** and ***n*** respectively. Write a C program to find the median of the array obtained after merging the above two arrays. Assume that $1 \leq m \leq 100$ and $1 \leq n \leq 100$.

Your program should implement the following functions.

main() - repeatedly reads a character ***'r', 'f'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.

read(n) – read ***n*** integers from the terminal and store it in an array.

findMedian(X, Y, m, n) – find the median of the array after, merging the elements of ***Y*** of size ***n*** with the elements of ***X*** of size ***m***.

display(A) – display the elements of ***A***.

INPUT FORMAT:

First line contains a character from {***'r', 'f', 'd' ,'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of space separated ***n*** integers, to be read into the array.

• Character ***'f'*** is to find the median value of the given two arrays after merging the arrays.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT:

For option ***'f'***, print the median value from the two given arrays after merging.

For option ***'d'***, print the elements of array separated by space in a new line.

Sample Input:

```
r 1
900
r 4
5 8 10 20
f
t
```

Sample Output:

```
10
```

6. Given an array ***A*** of ***n*** characters, sort the array in ascending order using heap sort. Assume that $1 \leq n \leq 100$.

Your program should implement the following functions:

main() - repeatedly reads a character ***'r', 's'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.
read(n) – read ***n*** characters from the terminal and store it in an array ***A***.

heapSort(A) – perform heap sort on array ***A.*** Use display() function to print the sorted array at the end of the function.

display(A) – display the elements of ***A***.

INPUT FORMAT:

First line contains a character from {***'r', 's', 'd' ,'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of ***n*** characters separated by a space, to be read into the array.

• Character ***'s'*** is to sort the given array using heap sort.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT:

For option ***'s'***, print the output for heap sort on the given character array.

For option ***'d'***, print the elements of array separated by space in a new line.

Sample Input:

```
r 7
A P M N D B C
d
s
t
```

Sample Output:

```
A P M N D B C
A B C D M N P
```

7. You are given an unsorted array ***X*** of integers of size ***m***, and another integer ***k***. Write a C program to find ***k****th* smallest element in the array using **maxheap**. If ***k*** > ***n***, then **k = k (modulo n)**. Your program should implement the following functions.

main() - repeatedly reads a character ***'r'***, ***'s'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.

read(n) – read ***n*** integers from the terminal and store it in an array ***X***.

kthSmallest(X, m, k) – find the ***k****th* smallest element from array ***X*** of size ***m*** using ***maxheap***.

display(A) – display the elements of ***A***.

INPUT FORMAT:

First line contains a character from {***'r', 's', 'd' ,'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of space separated ***n*** integers, to be read into the array.

• Character ***'s'*** is followed by an integer ***k***, and find the ***k****th* smallest element from array.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT:

For option ***'s'***, print the ***k****th* smallest element in the given array.

For option ***'d'***, print the elements of the array separated by space in a new line.

Sample Input:

```
r 6
7  10  4  3  20 15
d
s 3
t
```

Sample Output:

```
7  10  4  3  20 15
7
```

8. Consider an unsorted array ***X*** of integers with size ***n*** and another integer ***k***. Write a C program to find ***k**th*** largest element in the given integer array using **partition()** method of **quicksort**. If ***k*** > ***n***, then **k = k (modulo n)**.

Your program should implement the following functions.

main() - repeatedly reads a character ***'r', 's'*** or ***'d'*** from the terminal and calls the sub-functions appropriately until character ***'t'*** is entered.

read(n) – read ***n*** integers from the terminal and store it in an array ***X***.

kthLargest(X, m, k) – find the ***k*** th largest element from array ***X*** of size ***m*** using **partition()** method of **quicksort**.

display(A) – display the elements of ***A***.

INPUT FORMAT:

First line contains a character from {***'r', 's', 'd' ,'t'***} followed by zero or one integer.

• Character ***'r'*** is followed by an integer ***n***, the size of the input array. In this case, the second line gives the list of space separated ***n*** integers, to be read into the array.

• Character ***'s'*** is followed by an integer ***k***, and find the ***k*** th largest element from array.

• Character ***'d'*** is to display the contents of array.

• Character ***'t'*** terminates the program.

OUTPUT FORMAT:

For option ***'s'***, print the ***k*** th largest element in the given array.

For option ***'d'***, print the elements of array separated by space in a new line.

<u>Sample Input</u>

```
r 6
7 10 4 3 20 15
d
s 3
t
```

<u>Sample Output</u>

```
7 10 4 3 20 15
10
```