Shiv Nadar Institution of Eminence, Delhi, NCR

Lab sheet for CSD101 (Introduction to computing and Programming)

Semester of Implementation: Monsoon, 2024

Instructors: Dr. Suchi Kumari (suchi.kumari@snu.edu.in), **Dr. Sweta Kumari** (sweta.kumari@snu.edu.in), **Dr. Sumit Shekhar** (sumit.shekhar@snu.edu.in)

TA: Mr. Bhanu Prakash (bhanu.prakash@snu.edu.in), **Mr. Mithun Kumar** (mithun.kumar@snu.edu.in)

Instructions:

1. Once you complete the assignment, please show it to the TA.
2. Students must come to the lab and must show the assignments in the designated lab hours. Day-to-day lab performances will be recorded and will carry 15% weightage in internal assessment.
3. Lab will start in exact time. Students should enter the lab and take a seat 5 minutes before.
4. It is recommended to use LINUX platform for execution of the program.
5. Batch change to show the assignments WILL NOT be allowed.
6. Malpractice (in ANY form) will attract heavy penalties.
7. A useful link: https://www.w3schools.com/c/index.php

Lab Assignment 10

Deadline: 10-11-2024 (11:55 PM) for Monday batch

12-11-2024 (11:55 PM) for Wednesday batch

13-11-2024 (11:55 PM) for Thursday batch

14-11-2024 (11:55 PM) for Friday batch

Total Marks: 100

Objective: Programs based on Sorting and Searching

Steps to run C program

Step 1: gedit filename.c

Step 2: Compiling using GCC compiler

We use the following command in the terminal for compiling our filename.c source file

$ gcc filename.c –o filename

Step 3: Executing the program

After compilation executable is generated and we run the generated executable using the below command.

$ ./filename

Q1. Sorting Challenge: The Great Sorting Duel!

Attention, aspiring coders! Prepare for a thrilling showdown between two classic sorting champions: **Selection Sort** and **Insertion Sort.** Your quest is to develop a C program that pits these two algorithms against each other. You'll wield a mystical array and gather data to reveal the true sorting sorcerer!

Your Epic Task:
1. *Implement the Sorting Spells*: Code both Selection Sort and Insertion Sort—let the algorithms shine. Fill in the code for functions *selectionSort()* and *insertionSort()* in the provide **sort_battle.c** file.
3. *Time Your Duels*: Measure which sorting method completes the task in record time. Change the size of the array for different time tests, keep it as 10, 100, 1000, 10000.
4. *Share Your Findings*: Analyze your results! Who emerged victorious in the battle of sorting?

Input and Output

n = 10
Selection Sort Time: 0.000002 seconds
Insertion Sort Time: 0.000002 seconds

n = 100
Selection Sort Time: 0.000018 seconds
Insertion Sort Time: 0.000007 seconds

n = 1000
Selection Sort Time: 0.001218 seconds
Insertion Sort Time: 0.000723 seconds

Q2. Search Party Challenge: Linear vs. Binary!

Attention, future tech wizards! Are you ready to embark on a thrilling adventure through the world of searching algorithms? Your mission is to craft a C program that allows you to pit **Linear Search** against **Binary Search** in a battle for efficiency!

Here's what you need to do:

1. *Create a Treasure Map (Array)*: Generate a sorted array filled with random numbers—like treasures waiting to be discovered. Fill in the *generateSortedArray*() function with sorting code based on "bubble-sort" in the provided **search_compare.c** file.

2. *Implement Two Search Heroes*: Code up Linear Search and Binary Search and prepare them for action. Fill in the code for *linearSearch*() and *binarySearch*() functions.

4. *Time the Search*: Measure how long each search method takes to find (or not find) the treasure! Analyze as to which search method was faster for different array sizes?

Input and Output

Enter the number of elements (up to 10000): 1000

Enter the target value to search: 456

Linear Search: Not found

Time taken for Linear Search: 0.000004 seconds

Binary Search: Not found

Time taken for Binary Search: 0.000001 seconds

Enter the number of elements (up to 10000): 10000

Enter the target value to search: 3456

Linear Search: Not found

Time taken for Linear Search: 0.000020 seconds

Binary Search: Not found

Time taken for Binary Search: 0.000001 seconds

Complementary Assignment for self-practice

Q3. You already know how to write a program for sorting an array using the Selection Sort algorithm. Extend the program with a function that counts and returns the number of swaps made.

Input and Output:

Enter the array elements: 64, 25, 12, 22, 11

Sorted array: 11, 12, 22, 25, 64

Swaps made: 3

Q4. Modify the linear search function in the second question to also count the number of comparisons made during the search.

Input and Output

Enter array: 5, 2, 4, 6, 1, 3

Enter the element to be found: 4

Element found at index: 2

Comparisons made: 3

Submission Format:- You have to upload: (1) The source code in the following format in a zipped folder: Assgn10_RollNo.zip. Inside the zipped folder save each program with Assgn6_task#_RollNo.c

Note: Please follow this naming convention mentioned above.

Grading Policy:- The policy for grading this assignment will be - (1) show to TA 66 marks (2) Code submission with indentation: 34 marks.

- All submissions are subject to plagiarism checks. Any case of plagiarism will be dealt with severely.