**Programming - Sorting and Searching**

## Solve problem A OR B

You can use any programming language.

**A) Selection Sort**

A Selection Sort searches an array looking for the smallest element in the array, then swaps this element with the first element of the array. The process is repeated for the subarray beginning with the second element. Each pass of the array places one element in its proper location. Write a method to implement Selection Sort, assuming an array of 10 integers as input. What are the minimum and maximum number of (a) comparisons and (b) swaps that Selection Sort can make in sorting a sequence of n elements. A subarray is defined as a range of elements within an array defined by a beginning index and an ending index within an array.

**B) Binary Search**

Write a program to implement Binary Search. Binary search can only be performed on a sorted array. The strategy is to compare an input element with the median element of the given array. If the input element is greater than the median element then the same procedure is repeated for a sub-array to the left of the median element, else the same procedure is repeated for the sub array to the right of the median element. The procedure is continued till the element is found or the search has exhausted. If there are 10 elements the median element is 5. A subarray is defined as a range of elements within an array defined by a beginning index and an ending index within an array. Find the maximum number of comparisons done in order to find an element in an array of 1000 elements.

**Programming - Recursion**

## Solve problem A OR B

You can use any programming language.

**A) Greatest Common Divisor**

The greatest common divisor of two integers is the largest integer that evenly divides each of the two numbers. Write a program that **recursively** calculates the greatest common divisor of two numbers.

**B) Permutations**

Write a **recursive** function that prints all the permutations of the first n characters of a string.

For example the call print("ABC", 3) would print

ABC

ACB

BAC

BCA

CBA

CAB

**Object Oriented Design and Programming**

**Card Class Implementation**

Specify design and implement a class for a card in a deck of playing cards. The class should store suit and rank of the card as member variables. Implement all appropriate methods including constructor(s) with correct scopes. Note that the class implementation must conform to standards of object-oriented programming.

You can use C++/Java/Ruby

**SQL**

### Budget Versus Actual Problem

The nature of this problem is to provide a query whose result helps you track broad budget category amounts to more finely defined actual task entries when money is spent.

Consider the two table with the following structure and data

## Budget

|  |  |  |
| --- | --- | --- |
| **Task** | **Category** | **Estimated\_Cost** |
| 1 | 9100 | 100.00 |
| 2 | 9100 | 15.00 |
| 3 | 9100 | 6.00 |
| 4 | 9200 | 8.00 |
| 5 | 9200 | 11.00 |

## Actuals

|  |  |  |
| --- | --- | --- |
| **Voucher** | **Task** | **Actual\_Cost** |
| 1 | 1 | 10.00 |
| 2 | 1 | 20.00 |
| 3 | 1 | 15.00 |
| 4 | 2 | 32.00 |
| 5 | 4 | 8.00 |
| 6 | 5 | 3.00 |
| 7 | 5 | 4.00 |

Create a query that will provide a comparison between the estimated and actual spending by category

|  |  |  |
| --- | --- | --- |
| **Category** | **Estimated\_Cost** | **Actual\_Cost** |
| 9100 | 121.00 | 77.00 |
| 9200 | 19.00 | 15.00 |

Implement the solution using (a) a view and a query (b) a single query

**Data Modeling**

**Create a data model to store the following chart**

