**IE-MCT-Task-Phase**

**Questionnaire Solutions**

**Name** – Ansh Goyal

**Mentor Name** - Yeshwant

Link to my **Github Repository** for codes - <https://github.com/aghogwarts/IE-MCT-Task-Phase>

**Theory**

* **Q1. How to search a missing number in an array that contains integers from 1 to 100?**

First we either create a hardcoded array or make one with user input.

Then we make a for loop which iterates through the numbers 1-100. Inside the loop we create another for loop to iterate through every number in the defined/procured array to check with the number from the outer for loop and print it if it doesn't match any of the values from the array.

* **Q2. How to delete the repeated elements in an integer array?**

We create a for loop to iterate through every value in the int array. We then check the iterated value through every element in the array and if it matches any duplicate value, we will store the position of the element.

Then we will shift all the elements to the right of the position found one place back by directly replacing `arr[i]` with `arr[i+1]`; simultaneously we will reduce the length of the array by 1 after every replacement.

* **Q3. Difference between float, double, long and int?**
* **int** is a data type used to store whole integers such as -45 or 5738
* **long** is a data type used to store longer values of integers when integers start losing their precision after a limit
* **float** is a data type used to store numbers with decimal values for ex 45.822 but with a precision of upto around 6-7 decimal places
* **double** is a data type used to store numbers with decimal values but with a precision of around 13 decimal places

**Program**

* **Q1. Implement the best sorting algorithm that you know & also explain why it is the best.**

I know only Binary sorting and Selection Sorting algorithms as of now. Of them two I think Selection sort is the best sorting algorithm as the number of iterations it performs to sort the array is lesser than that taken by Binary Sort.

*#include* <iostream>

using std::cin;

using std::cout;

int main()

{

    int i, j, n, pos, small, a[50];

    cout << "\nEnter length of the array: ";

    cin >> n;

    cout << "\nEnter elements of the array: \n";

*for* (i = 0; i < n; i++)

    {

        cin >> a[i];

    }

*for* (i = 0; i < n - 1; i++)

    { *// loop for number of pass*

        pos = i;

        small = a[i];

*for* (j = i + 1; j < n; j++)

        { *// loop for searching the smallest*

*if* (small > a[j])

            { *// finding the smallest*

                pos = j; *// pos for interchanging*

                small = a[j]; *// assigning current small value*

            }

        }

        a[pos] = a[i]; *// interchanging values*

        a[i] = small;

    }

    cout << "\n-- Sorted array is:\n>> ";

*for* (i = 0; i < n; i++)

    {

        cout << a[i] << "\t";

    }

    cout << "\n\n";

*return* 0;

}

* **Q2. Program to find the Maximum and minimum of two numbers without using any loop or condition.**

Directly use the inbuilt **max** and **min** functions

*#include* <iostream>

using namespace std;

int main()

{

    int a, b;

    cout << "\nType a number: ";

    cin >> a;

    cout << "Type another number: ";

    cin >> b;

    cout << "\nMaximum: " << max(a, b) << " || Minimum is: " << min(a, b) << "\n\n";

*return* 0;

}

* **Q3. Print a number 100 times without using loop and recursion.**

I tried using recursion to print it n number of times and used unsigned integer keyword hoping the iterations stop at 0 but they didn't and it's printing till infinity. So here's my attempt -

*#include* <iostream>

using namespace std;

int print(int a, unsigned int n)

{

    cout << a << "\n";

    n--;

*return* print(a, n);

} *// I tried using recursion and unsigned integer to make it stop at 0 and print it continuously without loop or conditions but it's still printing till infinity*

int main()

{

    int num;

    unsigned int times = 5;

    cout << "\nEnter a number: ";

    cin >> num;

    print(num, times);

*return* 0;

}