**Lab # 09 Working with Arrays and Pointers**

**Reading Task 1:**

**Working with Arrays**

Chapter 08 Arrays (pages 269 to 288) from the book: “Let us C” by Yashavant Kanetkar

**Reading Task 2: Selection Sort Algorithm**

In computer science, selection sort is a sorting algorithm, specifically an in-place comparison sort. It has O(n2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.

The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

**In-Lab Task 1:**

**Finding Minimum and Maximum Values in an Array**

Your task is to perform some functions on integer arrays. Specifically you will write a C program that does the following:

1. Declare an array of size 20.

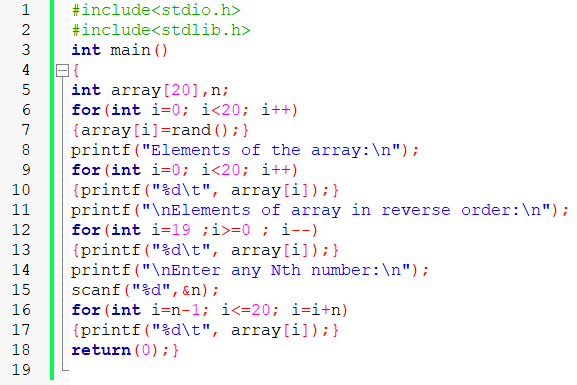
2. Initialize the array with random values (use loop, and rand() function).

3. Print all the elements in the array.

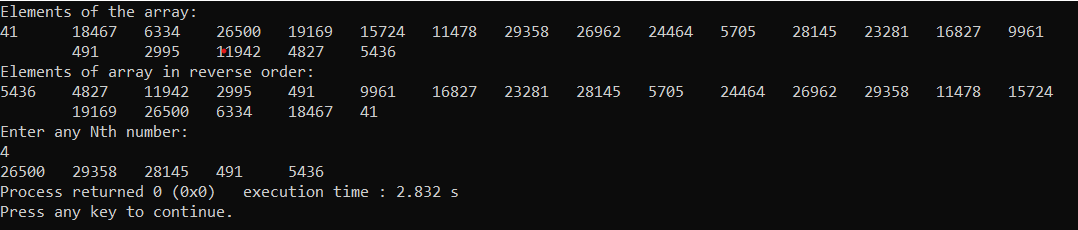
4. Print all the elements in the array in the reverse order.

5. Print the array such that every Nth element gets printed. N is user input.

**Program code:**



**Output:**



**In-Lab Task 2: Implementing Selection Sort**

You are given a C program in **Code Listing 1**, that does the following:

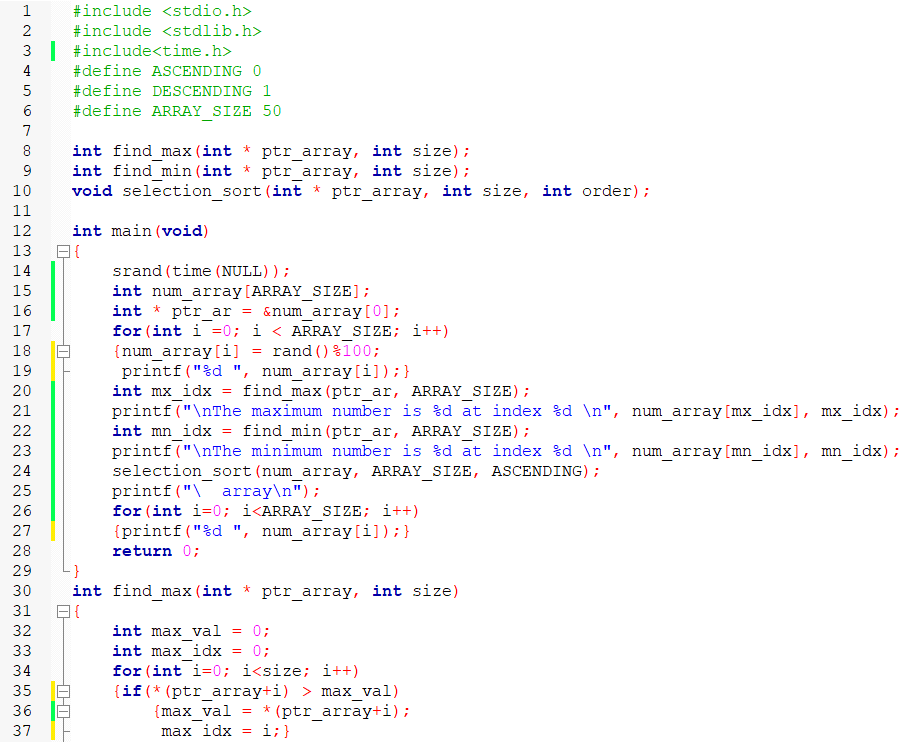
1. Declares an integer array with 50 elements (not initialized).

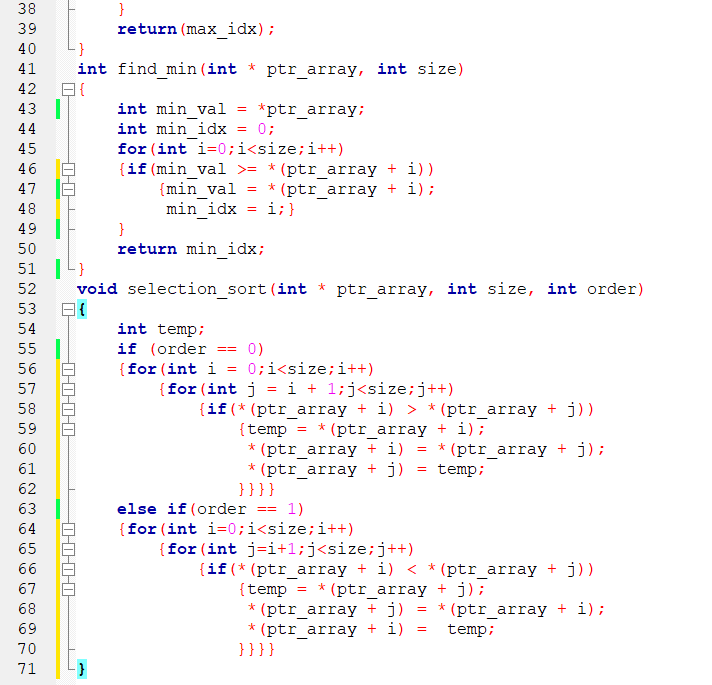
2. Populates the array with random positive numbers. (Uses a loop and rand() function)

3. Calls the function ‘int **find\_max**(int \* ptr\_array, int size)’ and prints the value and index of the largest number.

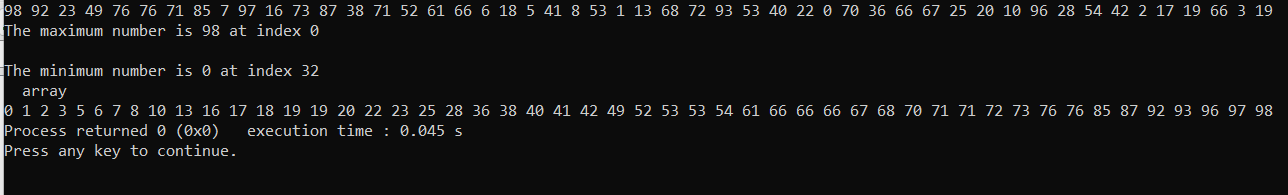
Write a similar function ‘int **find\_min**(int \* ptr\_array, int size)’ and print the value and index of the smallest number in the array as well.

**Program code:**





**Output:**



**Post-Lab Task:**

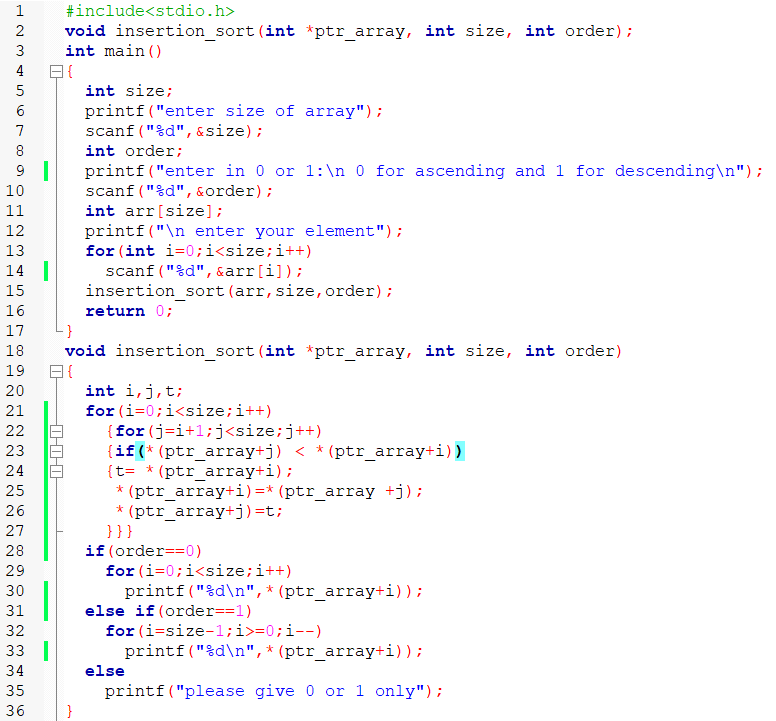
**Implement Insertion Sort Algorithm :**

Your second task is to implement the Insertion Sort algorithm by making a function with the following prototype;

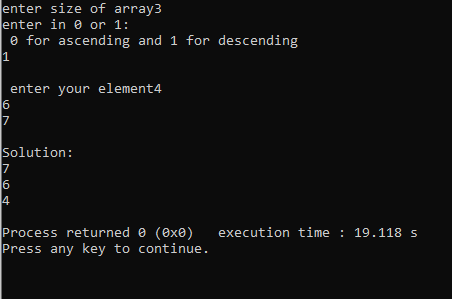
void insertion\_sort**(**int **\*** ptr\_array**,** int size**,** int order**);**

This function takes as input a pointer to the start of the array, and the array size and sorts it in-place. The last input to the function is the sorting order (0 for ascending and 1 for descending).

**Program code:**



**Output:**



**Critical Analysics:**