

Programming Fundamentals

Lab Report

Lab 09



Group Members Name & Reg #:	<u>Muhammad Haris Irfan</u> (FA18-BCE-090)
Class	Programming Fundamentals CSC103 (BCE-2B)
Instructor's Name	Dilshad Sabir

In Lab Tasks

Question no: 1

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.

Solution:

```
#include <stdlib.h>

int main()
{
    int n;
    int arr[20];
    int *ptr_marks= &arr[0];
    for(int i=0; i<20; i++)
    {

        *ptr_marks= rand();
        ptr_marks ++;

    }

    for(int i=0;i<20;i++)
    {
        printf("the value at address %d is %d\n",&arr[i],arr[i]);

    }

    printf("\n\n\n in reverse order\n\n");

    for(int i=19;i>=0;i--)
    {
        printf("the value at address %d is %d\n",&arr[i],arr[i]);

    }
}
```

```

printf("\n\n Enter the value of N?\n");
scanf("%d",&n);

printf("\n\nthe nth values are:\n");

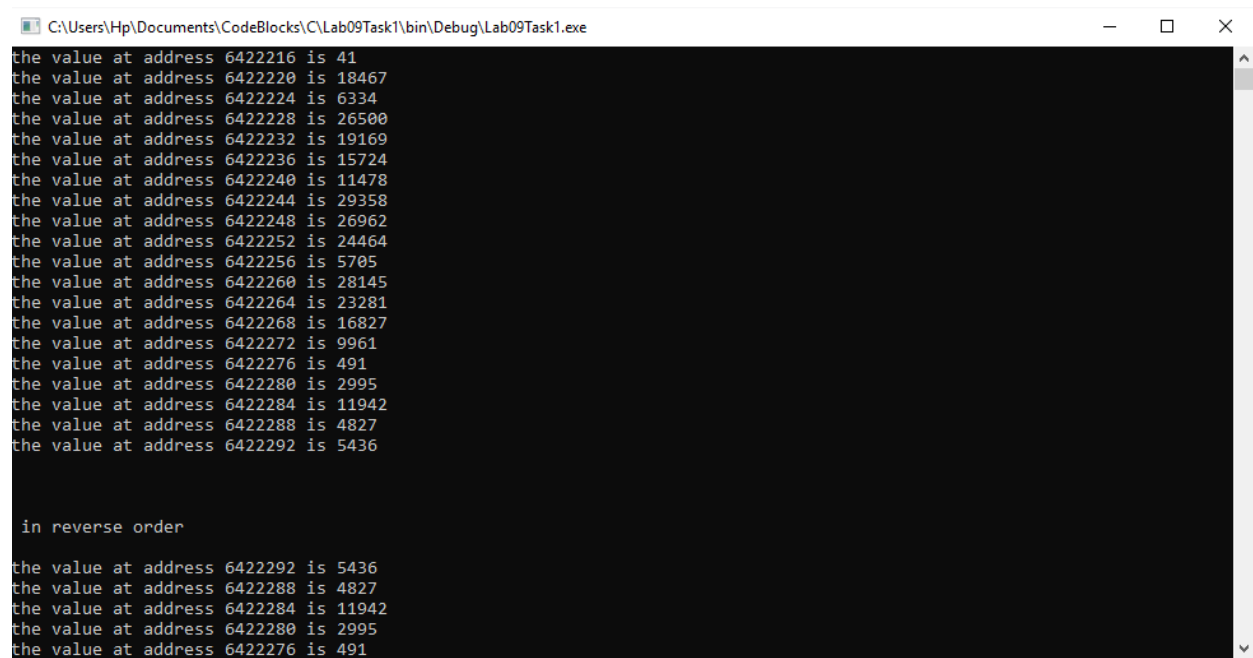
for(int i=0;i<20;i=i+n)
{
    printf("the value at address %d is %d\n",&arr[i],arr[i]);

}

}

```

The Result of the following code is attached below:

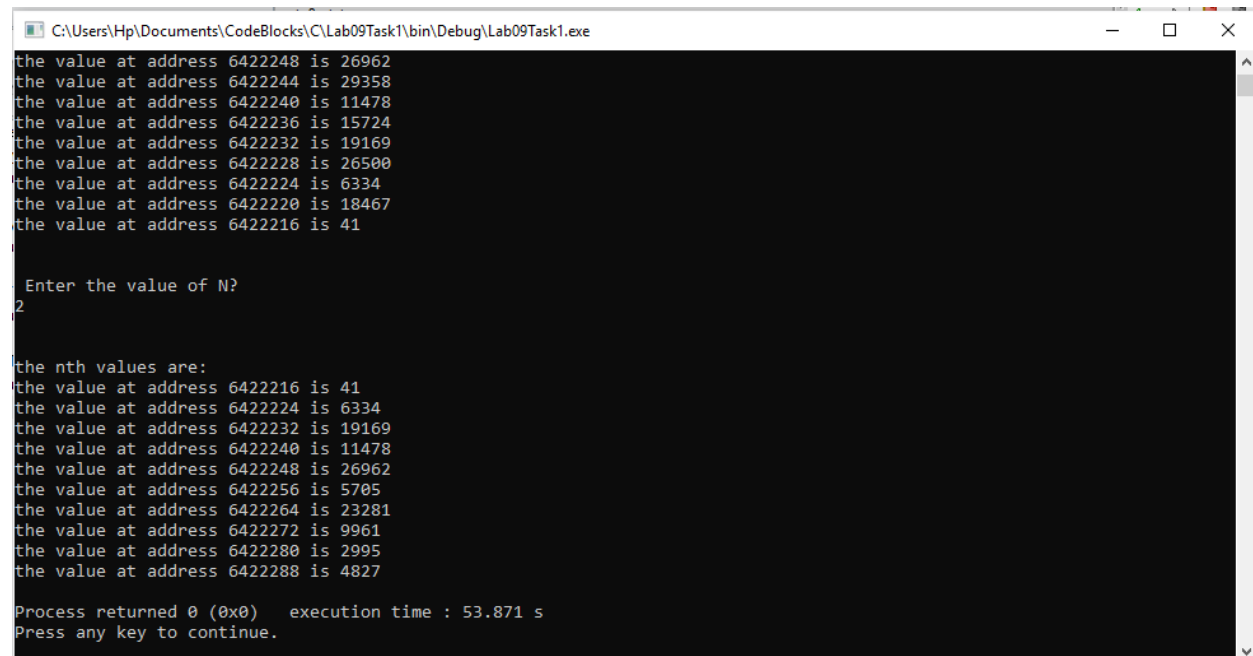


```

C:\Users\Hp\Documents\CodeBlocks\C\Lab09Task1\bin\Debug\Lab09Task1.exe
the value at address 6422216 is 41
the value at address 6422220 is 18467
the value at address 6422224 is 6334
the value at address 6422228 is 26500
the value at address 6422232 is 19169
the value at address 6422236 is 15724
the value at address 6422240 is 11478
the value at address 6422244 is 29358
the value at address 6422248 is 26962
the value at address 6422252 is 24464
the value at address 6422256 is 5705
the value at address 6422260 is 28145
the value at address 6422264 is 23281
the value at address 6422268 is 16827
the value at address 6422272 is 9961
the value at address 6422276 is 491
the value at address 6422280 is 2995
the value at address 6422284 is 11942
the value at address 6422288 is 4827
the value at address 6422292 is 5436

in reverse order
the value at address 6422292 is 5436
the value at address 6422288 is 4827
the value at address 6422284 is 11942
the value at address 6422280 is 2995
the value at address 6422276 is 491

```



```

C:\Users\Hp\Documents\CodeBlocks\C\Lab09Task1\bin\Debug\Lab09Task1.exe
the value at address 6422248 is 26962
the value at address 6422244 is 29358
the value at address 6422240 is 11478
the value at address 6422236 is 15724
the value at address 6422232 is 19169
the value at address 6422228 is 26500
the value at address 6422224 is 6334
the value at address 6422220 is 18467
the value at address 6422216 is 41

Enter the value of N?
2

the nth values are:
the value at address 6422216 is 41
the value at address 6422224 is 6334
the value at address 6422232 is 19169
the value at address 6422240 is 11478
the value at address 6422248 is 26962
the value at address 6422256 is 5705
the value at address 6422264 is 23281
the value at address 6422272 is 9961
the value at address 6422280 is 2995
the value at address 6422288 is 4827

Process returned 0 (0x0) execution time : 53.871 s
Press any key to continue.

```

The output result verifies that our code is correct.

=====

Question no: 2(A)

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.

Solution

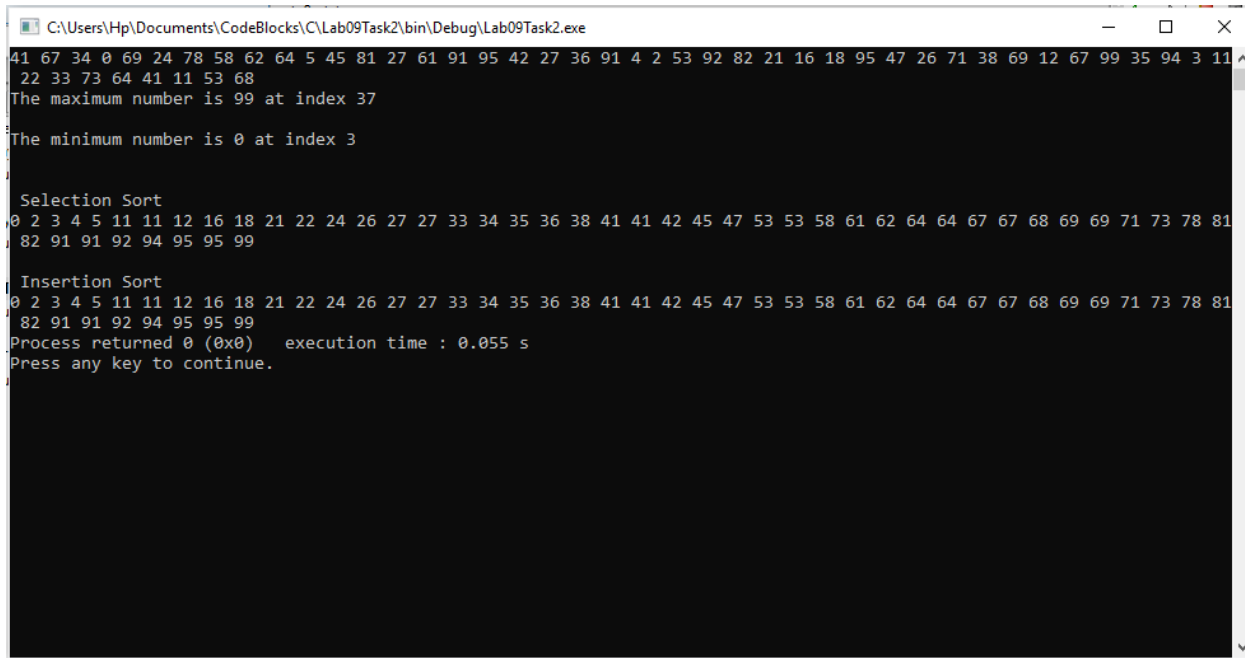
The code is shown below.

```
int find_min(int * ptr_array, int size)
{
    int min=9999;
    int index=0;

    for(int i=0; i<size ;i++)
    {
        /* ptr_array = &num_array[0];
        //int small=9999;
        //int compare=* ptr_array[i];
        //int index;
        if(min > *(ptr_array+i))

            {
                min =*(ptr_array+i) ;
                index=i;
            }
    }
    //printf("the smallest value is at index %d and is %d\n",index,min);
    return (index);
}
```

This code finds us the minimum number in the array.



```
C:\Users\Hp\Documents\CodeBlocks\C\Lab09Task2\bin\Debug\Lab09Task2.exe
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26 71 38 69 12 67 99 35 94 3 11
22 33 73 64 41 11 53 68
The maximum number is 99 at index 37
The minimum number is 0 at index 3

Selection Sort
0 2 3 4 5 11 11 12 16 18 21 22 24 26 27 27 33 34 35 36 38 41 41 42 45 47 53 53 58 61 62 64 64 67 67 68 69 69 71 73 78 81
82 91 91 92 94 95 95 99

Insertion Sort
0 2 3 4 5 11 11 12 16 18 21 22 24 26 27 27 33 34 35 36 38 41 41 42 45 47 53 53 58 61 62 64 64 67 67 68 69 69 71 73 78 81
82 91 91 92 94 95 95 99
Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.
```

The minimum value is shown; hence our program works.

=====

Question no: 2(B)

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

*void selection_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). You should call the functions (find_max(), find_min()) developed in lab-task 1 to implement Selection Sort algorithm.

Solution

The code is shown below.

```

void selection_sort(int * ptr_array, int size, int order)
{
    for (int c = 0; c < (size - 1); c++)
    {
        int position = c;

        for (int d = c + 1; d < size; d++)
        {
            if (*(ptr_array+position) > *(ptr_array+d))
                position = d;
        }
        if (position != c)
        {
            int swap = *(ptr_array+c);
            *(ptr_array+c) = *(ptr_array+position);
            *(ptr_array+position) = swap;
        }
    }
}

```

This code sorts the numbers in ascending order in the array.

```

C:\Users\Hp\Documents\CodeBlocks\C\Lab09Task2\bin\Debug\Lab09Task2.exe
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26 71 38 69 12 67 99 35 94 3 11
22 33 73 64 41 11 53 68
The maximum number is 99 at index 37
The minimum number is 0 at index 3

Selection Sort
0 2 3 4 5 11 11 12 16 18 21 22 24 26 27 27 33 34 35 36 38 41 41 42 45 47 53 53 58 61 62 64 64 67 67 68 69 69 71 73 78 81
82 91 91 92 94 95 95 99

Insertion Sort
0 2 3 4 5 11 11 12 16 18 21 22 24 26 27 27 33 34 35 36 38 41 41 42 45 47 53 53 58 61 62 64 64 67 67 68 69 69 71 73 78 81
82 91 91 92 94 95 95 99
Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.

```

hence our program works.

=====

Post Lab Task

Question:

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).

Solution

The code for this program is attached below,

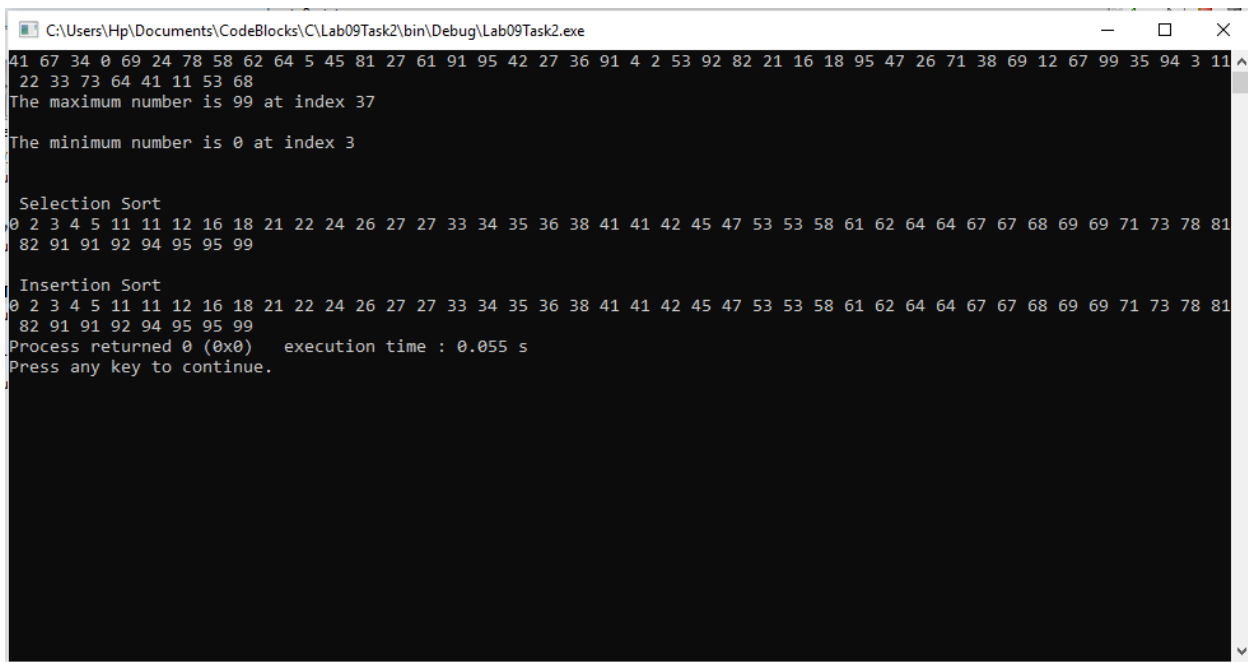
```
void insertion_sort(int * ptr_array, int size, int order)
{
    int c,d,t;

    for (c = 1 ; c <= size - 1; c++) {
        d = c;

        while ( d > 0 && *(ptr_array+d-1) > *(ptr_array+d)) {
            t= *(ptr_array+d);
            *(ptr_array+d)= *(ptr_array+d-1);
            *(ptr_array+d-1)= t;

            d--;
        }
    }
}
```

The Result of this program is attached below,



```
C:\Users\Hp\Documents\CodeBlocks\C\Lab09Task2\bin\Debug\Lab09Task2.exe
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26 71 38 69 12 67 99 35 94 3 11
22 33 73 64 41 11 53 68
The maximum number is 99 at index 37
The minimum number is 0 at index 3

Selection Sort
0 2 3 4 5 11 11 12 16 18 21 22 24 26 27 27 33 34 35 36 38 41 41 42 45 47 53 53 58 61 62 64 64 67 67 68 69 69 71 73 78 81
82 91 91 92 94 95 95 99

Insertion Sort
0 2 3 4 5 11 11 12 16 18 21 22 24 26 27 27 33 34 35 36 38 41 41 42 45 47 53 53 58 61 62 64 64 67 67 68 69 69 71 73 78 81
82 91 91 92 94 95 95 99
Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.
```

This result verifies that our code is correct as the we have sorted the array in ascending order using insertion sort.

THE END