Muhammad Rafi Cahya Ramadhana

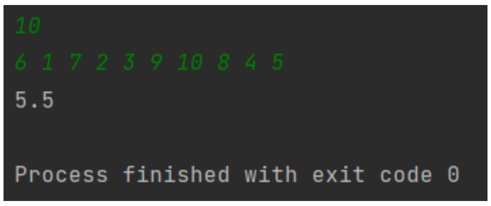
22/492162/PA/21075

WEEK 9 ASSIGNMENT

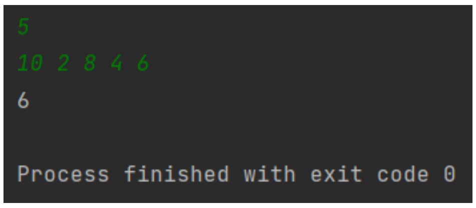
1. Source code:

```
#include <iostream>
using namespace std;

void insertArray(int arr[], int n){
    for (int i=0; i<n; i++){
        cin >> arr[i];
    }
}

void sortArray(int arr[], int n){
    for (int i=0; i<n; i++){
        for (int j=i+1; j<n; j++){
            if (arr[j] < arr[i]){
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

void medianArray(int arr[], int n){
    float median;
    if (n%2 == 0){
        median = (arr[((n/2)-1)] + arr[(n/2)]) * 1.0 / 2;
    } else {
        median = arr[(n/2)];
    }

    cout << median << endl;
}

int main() {
    int n;
    cin >> n;
    int arr[n];

    insertArray(arr, n);
    sortArray(arr, n);
    medianArray(arr, n);
}
```

Screenshot:



```
10
6 1 7 2 3 9 10 8 4 5
5.5

Process finished with exit code 0
```



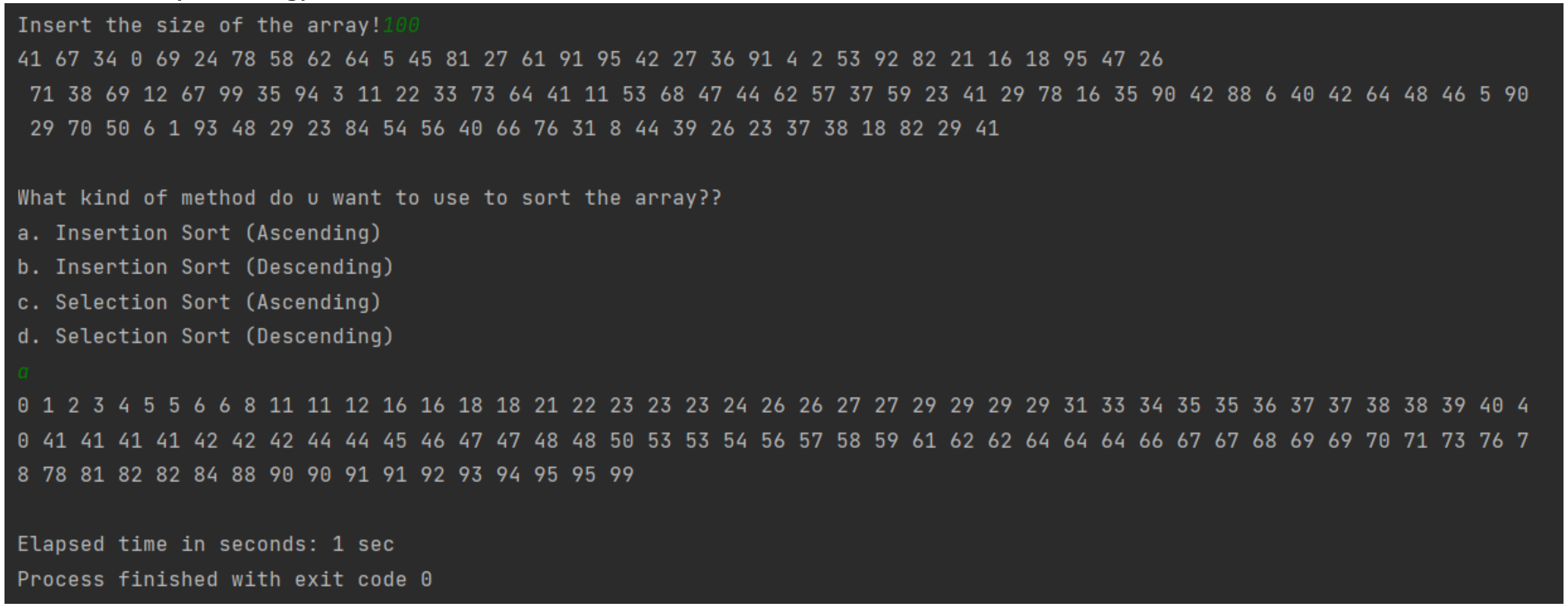
```
5
10 2 8 4 6
6

Process finished with exit code 0
```

2. Source code:

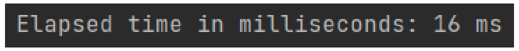
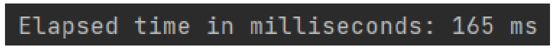
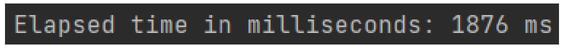
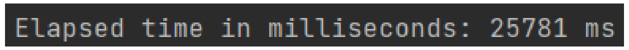
```
#include <iostream>
#include <chrono>
using namespace std;

void insertArray(int arr[], int n){
    for (int i=0; i<n; i++){
        cin >> arr[i];
    }
}

void displayArray(int arr[], int n){
    for (int i=0; i<n; i++){
        cout << arr[i] << " ";
    }
    cout << endl;
}

void insertionSortAscending (int arr[], int n){
    for (int j=1; j<n; j++){
        int i = j-1;
        int temp = arr[j];
        while(arr[i] > temp && i >= 0){
            arr[i+1] = arr[i];
            i--;
        }
        arr[i+1] = temp;
    }
}

void insertionSortDescending (int arr[], int n){
    for (int j=1; j<n; j++){
        int i = j-1;
        int temp = arr[j];
        while(arr[i] < temp && i >= 0){
            arr[i+1] = arr[i];
            i--;
        }
        arr[i+1] = temp;
    }
}

void selectionSortAscending(int arr[], int n){
    for (int i=0; i<n; i++){
        int min = i;
        for (int j=1+i; j<n; j++){
            if (arr[j] < arr[min]){
                min = j;
            }
        }
        int temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}

void selectionSortDescending(int arr[], int n){
    for (int i=0; i<n; i++){
        int min = i;
        for (int j=1+i; j<n; j++){
            if (arr[j] > arr[min]){
                min = j;
            }
        }
        int temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}

void randomizeArray(int arr[], int n){
    for (int i=0; i<n; i++){
        arr[i] = rand()%100;
    }
}

int main() {
    auto start = chrono::steady_clock::now();
    int n;
    cout << "Insert the size of the array!";
    cin >> n;
    int arr[n];

    randomizeArray(arr, n);
    displayArray(arr, n);

    cout << endl << "What kind of method do u want to use to
sort the array??" << endl << "a. Insertion Sort (Ascending)" <<
endl << "b. Insertion Sort (Descending)" << endl << "c.
Selection Sort (Ascending)" << endl << "d. Selection Sort
(Descending)" << endl;
    char x;
    cin >> x;
    switch (x) {
        case 'a':
            insertionSortAscending(arr,n);
            break;
        case 'b':
            insertionSortDescending(arr,n);
            break;
        case 'c':
            selectionSortAscending(arr,n);
            break;
        case 'd':
            selectionSortDescending(arr,n);
            break;
    }

    displayArray(arr, n);

    auto end = chrono::steady_clock::now();
    cout << endl << "Elapsed time in seconds: " <<
chrono::duration_cast<chrono::seconds>(end - start).count() <<
" sec";

}
```

Notes: There will be slight differences in time used because I manually input the number n in the Screenshot section, whereas n is predefined in code in the comparison table.

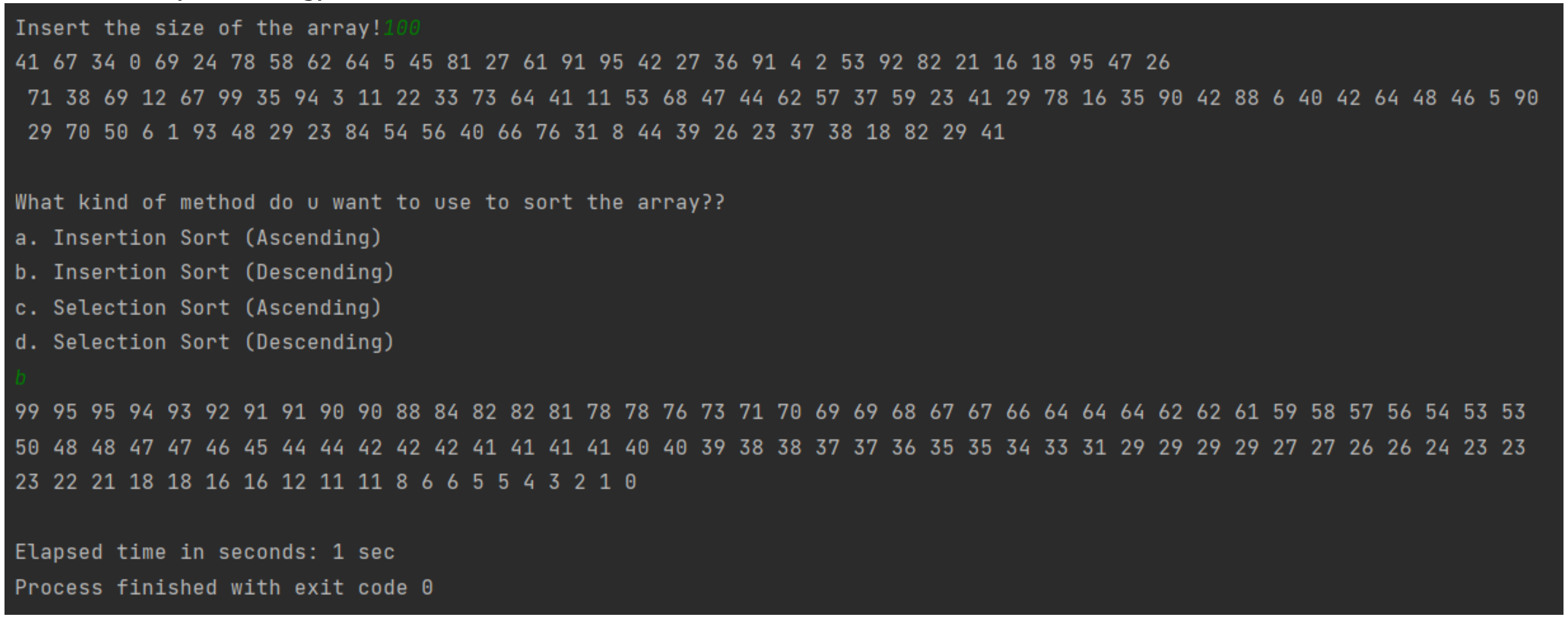
a. Insertion Sort (Ascending):



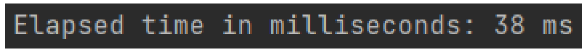
```
Insert the size of the array!100
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26
 71 38 69 12 67 99 35 94 3 11 22 33 73 64 41 11 53 68 47 44 62 57 37 59 23 41 29 78 16 35 90 42 88 6 40 42 64 48 46 5 90
 29 70 50 6 1 93 48 29 23 84 54 56 40 66 76 31 8 44 39 26 23 37 38 18 82 29 41

What kind of method do u want to use to sort the array??
a. Insertion Sort (Ascending)
b. Insertion Sort (Descending)
c. Selection Sort (Ascending)
d. Selection Sort (Descending)
a
0 1 2 3 4 5 5 6 6 8 11 11 12 16 16 18 18 21 22 23 23 23 24 26 26 27 27 29 29 29 29 31 33 34 35 35 36 37 37 38 38 39 40 4
0 41 41 41 41 42 42 42 44 44 45 46 47 47 48 48 50 53 53 54 56 57 58 59 61 62 62 64 64 64 66 67 67 68 69 69 70 71 73 76 7
8 78 81 82 82 84 88 90 90 91 91 92 93 94 95 95 99

Elapsed time in seconds: 1 sec
Process finished with exit code 0
```

- 100 Data 
  ```
  Elapsed time in milliseconds: 16 ms
  ```
- 1000 Data 
  ```
  Elapsed time in milliseconds: 165 ms
  ```
- 10000 Data 
  ```
  Elapsed time in milliseconds: 1876 ms
  ```
- 100000 Data 
  ```
  Elapsed time in milliseconds: 25781 ms
  ```

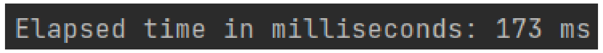
b. Insertion Sort (Descending):



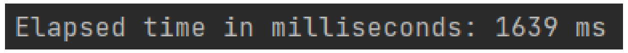
```
Insert the size of the array!100
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26
 71 38 69 12 67 99 35 94 3 11 22 33 73 64 41 11 53 68 47 44 62 57 37 59 23 41 29 78 16 35 90 42 88 6 40 42 64 48 46 5 90
 29 70 50 6 1 93 48 29 23 84 54 56 40 66 76 31 8 44 39 26 23 37 38 18 82 29 41

What kind of method do u want to use to sort the array??
a. Insertion Sort (Ascending)
b. Insertion Sort (Descending)
c. Selection Sort (Ascending)
d. Selection Sort (Descending)
b
99 95 95 94 93 92 91 91 90 90 88 84 82 82 81 78 78 76 73 71 70 69 69 68 67 67 66 64 64 64 62 62 61 59 58 57 56 54 53 53
50 48 48 47 47 46 45 44 44 42 42 42 41 41 41 41 40 40 39 38 38 37 37 36 35 35 34 33 31 29 29 29 29 27 27 26 26 24 23 23
23 22 21 18 18 16 16 12 11 11 8 6 6 5 5 4 3 2 1 0

Elapsed time in seconds: 1 sec
Process finished with exit code 0
```

- 100 Data 

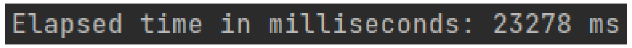
  ```
  Elapsed time in milliseconds: 38 ms
  ```

- 1000 Data 

  ```
  Elapsed time in milliseconds: 173 ms
  ```

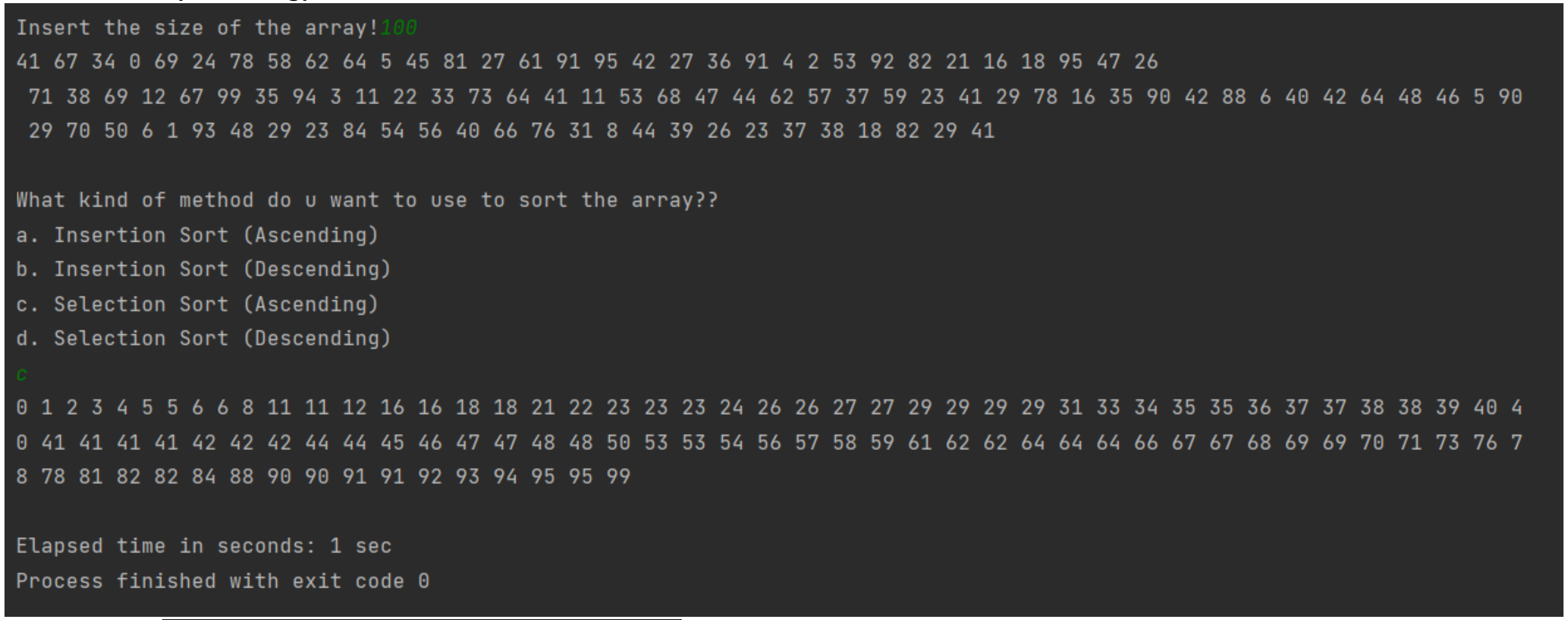
- 10000 Data 

  ```
  Elapsed time in milliseconds: 1639 ms
  ```

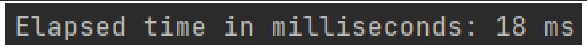
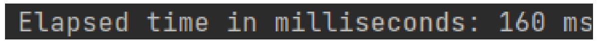
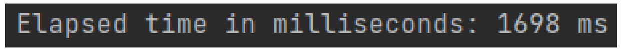
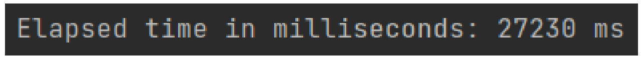
- 100000 Data 

  ```
  Elapsed time in milliseconds: 23278 ms
  ```

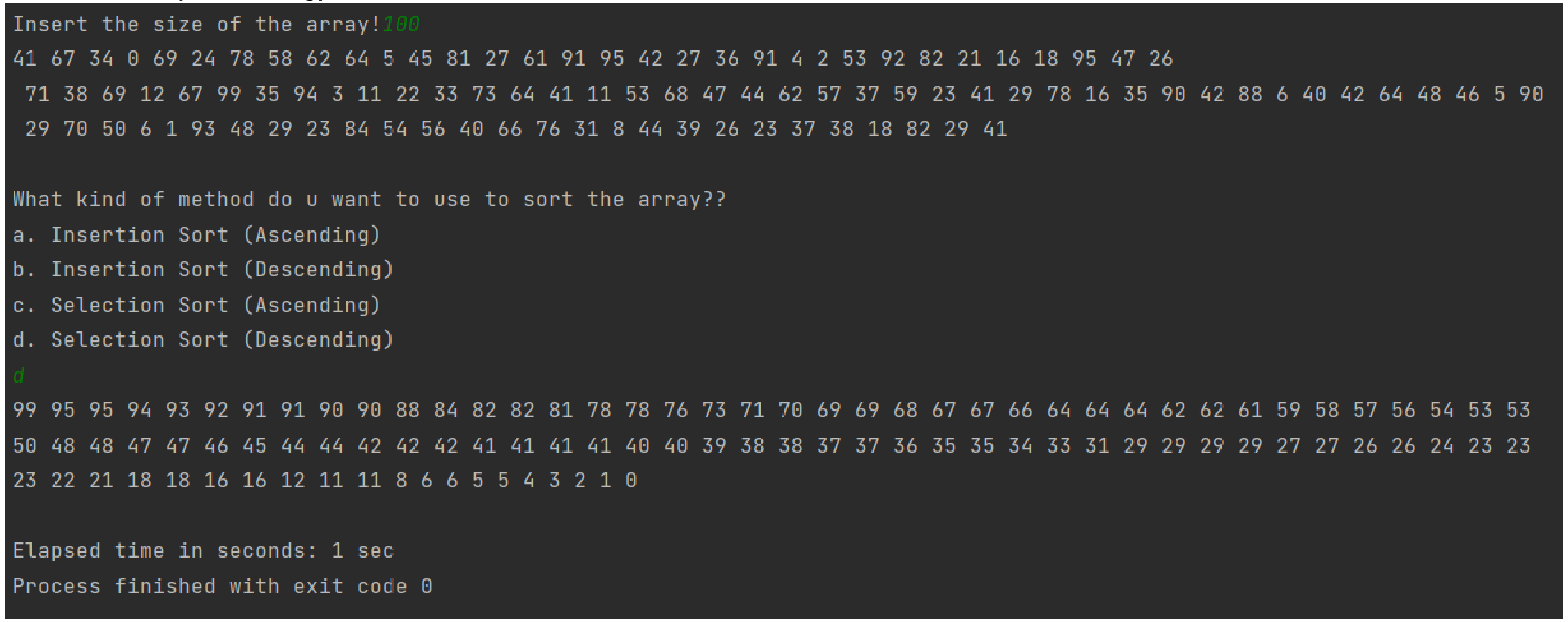
c. Selection Sort (Ascending):



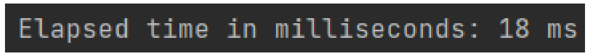
```
Insert the size of the array!100
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26
 71 38 69 12 67 99 35 94 3 11 22 33 73 64 41 11 53 68 47 44 62 57 37 59 23 41 29 78 16 35 90 42 88 6 40 42 64 48 46 5 90
 29 70 50 6 1 93 48 29 23 84 54 56 40 66 76 31 8 44 39 26 23 37 38 18 82 29 41

What kind of method do u want to use to sort the array??
a. Insertion Sort (Ascending)
b. Insertion Sort (Descending)
c. Selection Sort (Ascending)
d. Selection Sort (Descending)
c
0 1 2 3 4 5 5 6 6 8 11 11 12 16 16 18 18 21 22 23 23 23 24 26 26 27 27 29 29 29 29 31 33 34 35 35 36 37 37 38 38 39 40 4
0 41 41 41 41 42 42 42 44 44 45 46 47 47 48 48 50 53 53 54 56 57 58 59 61 62 62 64 64 64 66 67 67 68 69 69 70 71 73 76 7
8 78 81 82 82 84 88 90 90 91 91 92 93 94 95 95 99

Elapsed time in seconds: 1 sec
Process finished with exit code 0
```

- 100 Data 
  `Elapsed time in milliseconds: 18 ms`
- 1000 Data 
  `Elapsed time in milliseconds: 160 ms`
- 10000 Data 
  `Elapsed time in milliseconds: 1698 ms`
- 100000 Data 
  `Elapsed time in milliseconds: 27230 ms`

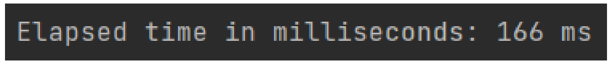
d. Selection Sort (Descending):



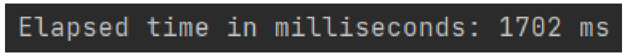
```
Insert the size of the array!100
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61 91 95 42 27 36 91 4 2 53 92 82 21 16 18 95 47 26
 71 38 69 12 67 99 35 94 3 11 22 33 73 64 41 11 53 68 47 44 62 57 37 59 23 41 29 78 16 35 90 42 88 6 40 42 64 48 46 5 90
 29 70 50 6 1 93 48 29 23 84 54 56 40 66 76 31 8 44 39 26 23 37 38 18 82 29 41

What kind of method do u want to use to sort the array??
a. Insertion Sort (Ascending)
b. Insertion Sort (Descending)
c. Selection Sort (Ascending)
d. Selection Sort (Descending)
d
99 95 95 94 93 92 91 91 90 90 88 84 82 82 81 78 78 76 73 71 70 69 69 68 67 67 66 64 64 64 62 62 61 59 58 57 56 54 53 53
50 48 48 47 47 46 45 44 44 42 42 42 41 41 41 41 40 40 39 38 38 37 37 36 35 35 34 33 31 29 29 29 29 27 27 26 26 24 23 23
23 22 21 18 18 16 16 12 11 11 8 6 6 5 5 4 3 2 1 0

Elapsed time in seconds: 1 sec
Process finished with exit code 0
```

- 100 Data 

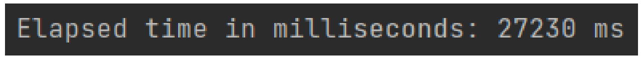
```
Elapsed time in milliseconds: 18 ms
```

- 1000 Data 

```
Elapsed time in milliseconds: 166 ms
```

- 10000 Data 

```
Elapsed time in milliseconds: 1702 ms
```

- 100000 Data 

```
Elapsed time in milliseconds: 27230 ms
```

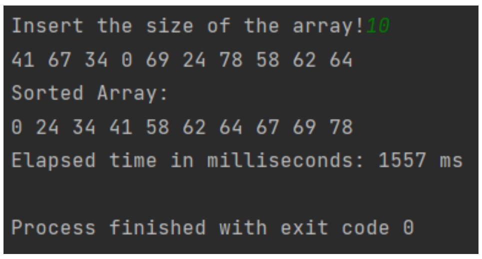
Time used comparison

	100 Data	1000 Data	10000 Data	100000 Data
Insertion Sort (Ascending)	16 ms	165 ms	1876 ms	25781 ms
Insertion Sort (Descending)	38 ms	173 ms	1639 ms	23278 ms
Selection Sort (Ascending)	18 ms	160 ms	1698 ms	27230 ms
Selection Sort (Descending)	18 ms	166 ms	1702 ms	27230 ms

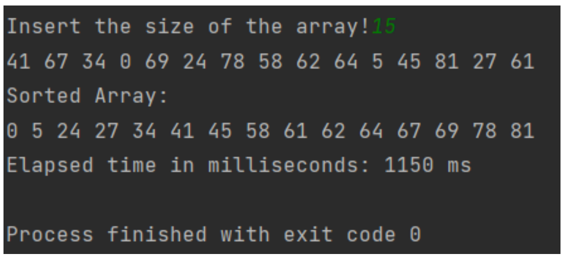
3. Source code:

```cpp
#include <iostream>
#include <chrono>
#include <algorithm>

using namespace std;

void displayArray(int arr[], int n){
    for (int i=0; i<n; i++){
        cout << arr[i] << " ";
    }
    cout << endl;
}

void randomizeArray(int arr[], int n){
    for (int i=0; i<n; i++){
        arr[i] = rand()%100;
    }
}

int main() {
    auto start = chrono::steady_clock::now();
    int n;
    cout << "Insert the size of the array!";
    cin >> n;
    int arr[n];

    randomizeArray(arr, n);
    displayArray(arr, n);

    sort(arr, arr+n);
    cout << "Sorted Array: " << endl;
    displayArray(arr, n);

    auto end = chrono::steady_clock::now();
    cout << "Elapsed time in milliseconds: "
         << chrono::duration_cast<chrono::milliseconds>(end -
start).count()
         << " ms" << endl;

}
```

Screenshot:



```
Insert the size of the array!10
41 67 34 0 69 24 78 58 62 64
Sorted Array:
0 24 34 41 58 62 64 67 69 78
Elapsed time in milliseconds: 1557 ms

Process finished with exit code 0
```



```
Insert the size of the array!15
41 67 34 0 69 24 78 58 62 64 5 45 81 27 61
Sorted Array:
0 5 24 27 34 41 45 58 61 62 64 67 69 78 81
Elapsed time in milliseconds: 1150 ms

Process finished with exit code 0
```

Notes: There will be slight differences in time used because I manually input the number n here, whereas n is predefined in code in the comparison table.

Time used comparison

	100 Data	1000 Data	10000 Data	100000 Data
Using sort() function	16 ms	176 ms	1696 ms	16731 ms

	100 Data	1000 Data	10000 Data	100000 Data
Insertion Sort (Ascending)	16 ms	165 ms	1876 ms	25781 ms
Insertion Sort (Descending)	38 ms	173 ms	1639 ms	23278 ms
Selection Sort (Ascending)	18 ms	160 ms	1698 ms	27230 ms
Selection Sort (Descending)	18 ms	166 ms	1702 ms	27230 ms

Conclusion: The result is pretty much similar in any given data except for 100000 data which the sort() function is comparably faster than any other sorting method used in the previous problem by 10000ish ms.