ASSIGNMENT 2

NAME :- KEVAL PANWALA
ROLL NO :- 21BCP041
GROUP :- G2

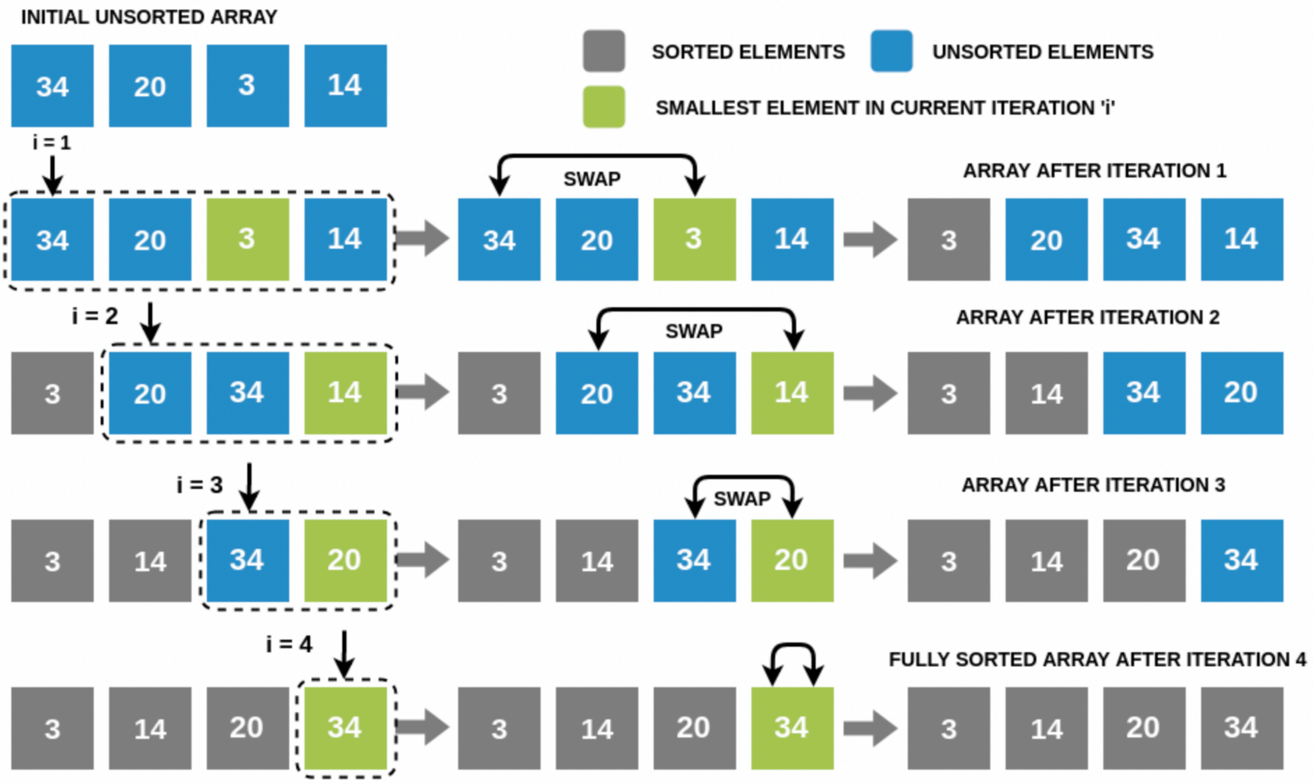
— — SELECTION SORT — —

○ AIM -

To write a program in which a user populated an unsorted array of integers and then it was sorted using selection sort. Finally, the program printed out the sorted array to the console.

○ INTRODUCTION -

Selection Sort Algorithm



- ALGORITHM -

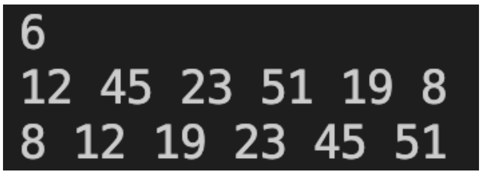
```
STEP 1 - Prompt the user for the size of the array.
STEP 2 - Create an array of integers of size given in step 1.
STEP 3 - For each element in the array.
            A.  Prompt the user to input a value.
            B.  Store that value as that element of the array.
STEP 4 - Set min to the first location.
STEP 5 - Search minimum element in the array.
STEP 6 - Swap the first location with the minimum value in the
           array.
STEP 7 - Assign the second element as min.
STEP 8 - Repeat the process until we get a sorted array.
STEP 9 - After that print the array using for loop.
```

- PROGRAM -

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char const *argv[])
{
    clock_t t, t1, t2;
    srand(time(0));
    int n;
    scanf("%d", &n);

    int array[n];
    for (int i = 0; i < n; i++)
    {
        array[i] = rand();
    }
    t = clock();
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            if (array[j]<array[i])
            {
                int temp = array[j];
                array[j] = array[i];
                array[i] = temp;
        }

    }

}
t = clock() - t;
double timet = ((double)t)/CLOCKS_PER_SEC;
for (int i = 0; i < n; i++)
{
    printf("%d ", array[i]);
}
printf("\n");
printf("Average time is %f\n", timet);

t1 = clock();
for (int i = 0; i < n - 1; i++)
{
    for (int j = i + 1; j < n; j++)
    {
        if (array[j]<array[i])
        {
            int temp = array[j];
            array[j] = array[i];
            array[i] = temp;
        }

    }

}
t1 = clock() - t1;
double timet_1 = ((double)t1)/CLOCKS_PER_SEC;
printf("Best time is %f\n", timet_1);

t2 = clock();
for (int i = 0; i < n - 1; i++)
{
    for (int j = i + 1; j < n; j++)
    {
        if (array[j]<array[i])
        {
            int temp = array[j];
            array[j] = array[i];
```

```
                    array[i] = temp;
                }

            }

        }
        t2 = clock() - t2;
        double timet_2 = ((double)t2)/CLOCKS_PER_SEC;
        printf("Worst time is %f\n", timet_2);
        return 0;
    }
```

- OUTPUT -



```
6
12 45 23 51 19 8
8 12 19 23 45 51
```

- ANALYSIS -

N	Best Case	Average Case	Worst Case
10	Time: 0.000001	Time: 0.000008	Time: 0.000002
100	Time: 0.000032	Time: 0.000062	Time: 0.000033
1000	Time: 0.001916	Time: 0.003467	Time: 0.002037
10000	Time: 0.062880	Time: 0.186224	Time: 0.064304
100000	Time: 6.288955	Time: 17.705892	Time: 6.308528

Worst Case Time Complexity -

If we want to sort in ascending order and the array is in descending order then, the worst case occurs.

Best Case Time Complexity -

It occurs when the array is already sorted.

Average Case Time Complexity -

It occurs when the elements of the array are in jumbled order.

Space Complexity -

Space complexity is `O(1)` because an extra variable `temp` is used.

- APPLICATIONS -

  The selection is used when:
  - A small list is to be sorted.
  - Cost of swapping does not matter.
  - Checking of all the elements is compulsory.
  - Cost of writing to a memory matters like in flash memory.

- REFERENCES -

  https://www.programiz.com/dsa/selection-sort