5. Best-case running time

a) Determine what the best-case running time of selection sort and describe it asymptotically.
b) Now, describe a simple modification of this selection sort algorithm so its best-case running time becomes Θ(n) for an input sequence of length n.

Answers:

a) We can consider the best case as when the input is an array where all the elements are same.
e.g. arr [ ] = {12,12,12,12,12,12}

The selection sort works as follows:

```
for (int i=0; i<n-1; i++) {
    min = i;                                  ... will execute for (n-1) times
    for (int j=i+1; j<n; j++) {
            if (arr[j]<arr[min])              ... this is explained below
                    min = j;                  ... won't execute at all (best case)
     }
     swap_numbers (&arr[i], &arr[min]);       ... will execute for (n-1) times
}
```

The if statement will:
execute (n-1) times if i = 0
execute (n-2) times if i = 1
execute (n-1-i) times for i (in general)

Total # of executions of 'if' statement = (n-1) + (n-2) + ... 1 = $\frac{n(n-1)}{2}$= $\frac{(n^2-n)}{2}$

To calculate the time complexity, let c1, c2 and c3 be the cost of execution for "first for loop", "second for loop" (i.e. if statement) and "swap_numbers function" respectively.

$$T(n) = c1*(n-1) + c2*\frac{(n^2-n)}{2} + c3*(n-1)$$

This equation is in form of $an^2 + bn + c$ with the n^2 term, which dominates all the other terms. Therefore, eliminating constants and other small terms, the best case running time of selection sort in asymptotic notation is: **O (n^2)**

b) For the best case to run in Θ(n), a possible modification can be done as follows in the above mentioned selection sort algorithm:

arr [ ] = {12, 12, 12, 12, 12, 12}

Check for the minimum element only once for i=0.
Skip over all the remaining elements if they are equal to the minimum element.

//Here, it will find min = 0 as for i = 0. Then, it will skip the elements which are equal to arr[min]. i.e. for i=1,2,3,4,5. It will reach at the end of the array and we get the sorted array without going for the next for loop. This will bring down the running time to Θ(n).

//But, for all the other cases, this modification doesn't matter as there won't be any same elements as such. e.g. {12,11,10,5,13}

```
for (int j=0; j<n; j++) {
    if(arr[j]<arr[min])                 ... will execute for (n-1) times
            min = j;                    ... constant time
}

int i=0;                                ... constant time
while(arr[i] == arr[min])
    i++;                                ... will execute for (n-1) times for
                                        the above mentioned best case

for (; i<n-1; i++) {
    min = i;
    for (int j=i+1; j<n; j++) {
        if(arr[j]<arr[min])
            min = j;
    }
    swap_numbers(&arr[i], &arr[min]);
}
```

The last for loop section will not run at all in the best case as the value of 'i' here will already be equal to n-1.

Thus, we got the selection sort's best case running time as Θ**(n)**.