Programming Tasks:

1. **Partition an array into two parts**

   Implement a method *Partition* declared below to divide a given array A into two parts. $j(j \geq 0)$ is the index of the pivot, put all elements $<$ pivot to the left, elements $\geq$ pivot to the right side.

```
template<typename T>
int Partition(array<T> A, int j,...){
    //...
}
```

2. **Non-randomized Quicksort**

   Implement a quick-sort algorithm *Quicksort* and take the last element in the array as pivot. Using *Partition* defined above as its subroutine.

```
template<typename T>
void Quicksort(array<T> A,...){
    //...
}
```

3. **Random Selection Algorithm**

   Implement a method *Rselect* to find i-th $(i > 0)$ smallest item of the given array A. Choose the pivot from A uniformly at random. The pseudocode is given below.

```
template<typename T>
T Rselect(array<T> A, int i, ....) {
    if only one element in A return A[0];
    Choose pivot p from array A uniformly at random;
    Partition A using pivot p;
    Let j be the index of p;
    if(j == i) return p;
    if(j > i) return Rselect(1st part of A,i,...);
    else return Rselect(2nd part of A,i-j,...);
}
```

4. **Deterministic Selection Algorithm**

   Implement a method *Dselect* using deterministic selection algorithm to find the i-th $(i > 0)$ smallest item of array A. The pivot is chosen by following steps

   (a) Break A into $n/5$ groups of size 5 each

   (b) Sort each group (e.g., use insertion sort, merge sort ...)

   (c) Copy $n/5$ medians into new array C

(d) Recursively compute median of C **by calling the deterministic selection algorithm**

(e) Return the median of C as pivot

The pseudocode of *Dselect* is as follows

```
template<typename T>
T Dselect(array<T> A, int i, ....) {
    if(only one element in A) return A[0];
    //ChoosePivot
    Break A into groups of 5, sort each group;
    C = n/5 medians;
    p = Dselect(C, n/10, ....);//return the median value in C

    //Partition
    Partition A using pivot p;
    Let j be the index of p;
    if(j == i) return p;
    if(j > i) return Rselect(1st part of A, i ,...);
    else return Rselect(2nd part of A, i-j-1 ,...);
}
```

5. **Complete main function**

   The *main.cpp* is given by us, try to follow the instructions in it to test your functions.

Important Notes:

- **Reference**: The class note of Lecture 7.
- Make sure your program won't crash if the input array is **empty**.
- Remember to submit your **makefile**!
- Due: **2019/10/26 11:59pm**