นายศิรภัทร โตตะหารก 6604062636348

Assignment 5 Divide and Conquer

1. พิจารณาอัลกอริทึม Quick Select ด้านล่าง

```
#include<stdio.h>
int arr[] = {1, 5, 10, 4, 8, 2, 6, 9, 20};
int k = 4;
int n = sizeof(arr) / sizeof(arr[0]);

int partition(int l, int r) {
     // **** use median of three for pivot selection

}

int quickSelect(int low, int high, int k) {

   if(low == high)
       return ..arr[low]....;

   int p = partition(arr, low, high);

    if (....p...k-1..........)        // case k = Pivot position
        return ...arr[p].......;
    else if (....p...>...k-1.......)                // case k ∈ L
        return quickSelect(arr, low, p-1, k);
    else {                                          // case k ∈ R
           k = .......p...........;
           return quickSelect(arr, p+1 , high , .....k..... );
         }
}

int main() {
    printf("%d", quickSelect(arr, ....0... , n-1 , k));
    return 0;
}
```

1.1 จงเติมโค้ดด้านบนเพื่อให้โปรแกรมทำงานได้สมบูรณ์

5

1.2 แสดงผลลัพธ์ในแต่ละรอบของการทำงาน (หลังจากทำ partition)

0 3 5

นายศิรภัทร โตเตะหารก 6604062636348

Assignment 5 Divide and Conquer

2. พิจารณา Pseudo code ด้านล่าง เพื่อหาสมาชิกในอาร์เรย์ A ที่มีค่าใกล้เคียงกับ M มากที่สุด K จำนวน

```
Algorithm K_nearest(A, M, k)
   Input: Array A, target M and k: number of nearest items
   Output: A[left+1] .... A[right-1] in k items

1. Sort A in ascending order
2. Find the index i in A closest to M
3. left = i-1, right = i
4. while (right - left - 1) < k do:
5.      print left, right
6.      if left < 0 then
7.           right = right + 1
8.       else if right >= n then
9.           left = left - 1
10.      else if abs(A[left]- M) > abs(A[right]- M) then
11.         right = right + 1
12. end algorithm
```

2.1 จงแสดงขั้นตอนการทำงานของอัลกอริทึม โดยใช้ข้อมูล A[] = {10, 12, 15, 17, 18, 20, 25} เมื่อ k = 2 และ M = 13

2.2 จงวิเคราะห์ time complexity ของอัลกอริทึมนี้ (ไม่รวมเวลา sorting **)

2.3 จงพัฒนาอัลกอริทึมดังกล่าวเพื่อแสดงผลลัพธ์ของ ข้อมูล A[] = {10, 12, 15, 17, 18, 20, 25} เมื่อ k = 4 และ M = 16

2.4 **(bonus)** จงปรับปรุงอัลกอริทึมดังกล่าว ให้ใช้เวลา **O(log n)** โดยใช้หลักการ divide and conquer