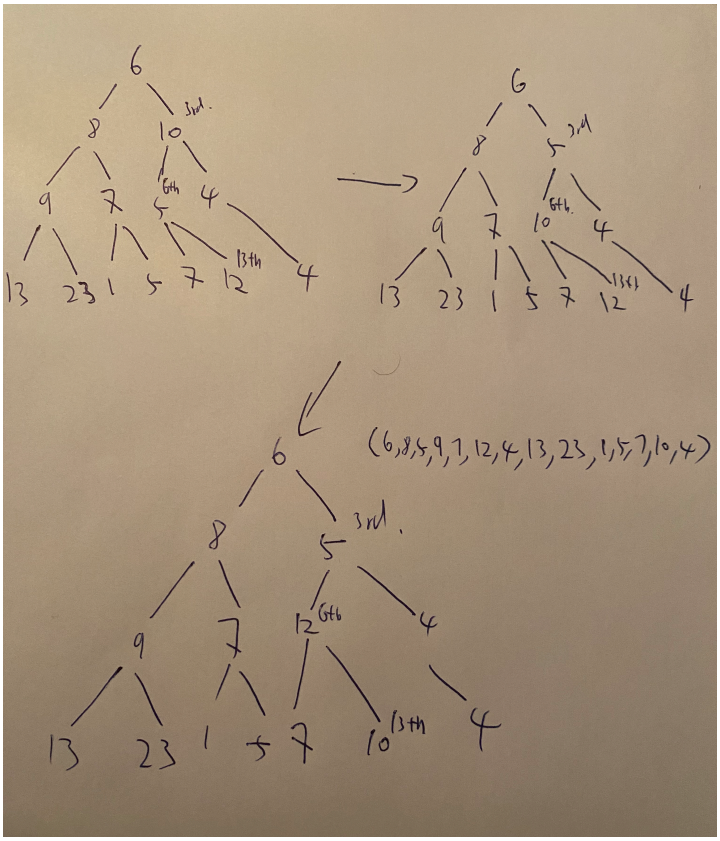
CS 566 HW3 Jixing Jacey Man

Question 1:

1. Max_Heap: a binary tree which value in each internal node is greater than or equal to the values in the children of that node.
    1. Index the array 6,8,10,9,7,5,4,13,23,1,5,7,12,4 from 1-14,start from the 3rd node to start swapping the 6th node (see hand draw picture)
       Left child 2*3 + 1 = 7
       Right child 2*3 + 1 = 8
       7+8 = 13, then swap with the 13th node
    2. The final array 6,8,5,9,7,12,4,13,23,1,5,7,10,4 there are total of 2 swap operation.

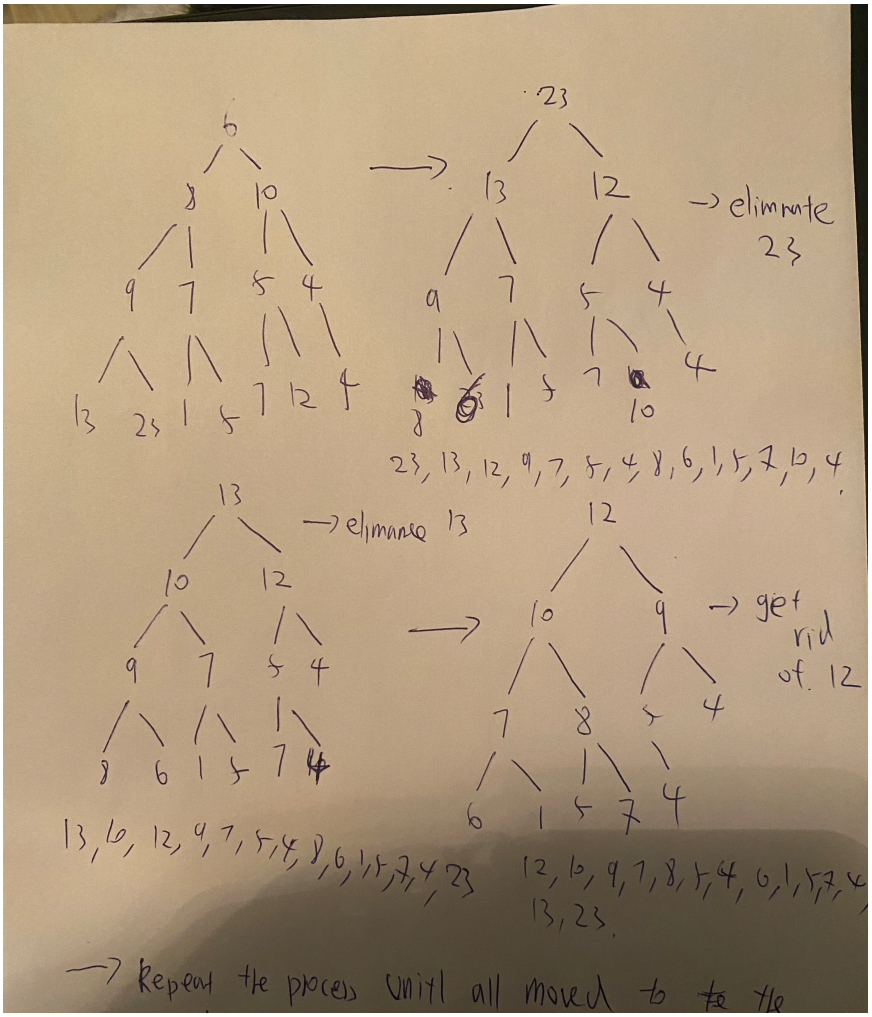


Question 2:

Run the HEAP-Sort Algorithm on the above array and count up the total number of exchanges. I am basically re-sorting the array from min to max
6,8,10,9,7,5,4,13,23,1,5,7,12,4 to 1,2,4,5,5,6,7,7,8,9,10,12,13,23

parent(i) = floor(i-1)/2
leftchild = 2*i+1
rightchild = 2*i+ 2

a. **write down the array after each key exchange** :see hand drawn picture and explanation below



So the idea is to re-sort the nodes from with each branch starting with the largest, once. Each time the largest number move to the top nodes, it is eliminated from the tree and pushed to the last index of the array. So 23, 13,12,9,7,5,4,8,6,1,5,7,10,4 when 23 the largest number is moved to the top of the tree, but then to 13,10,12,9,7,5,4,8,6,1,5,7,4,23 since the eliminated nodes is moved to the last index of the array. Repeat this process until eventually all nodes, except the smallest number(in this array, it is 1), and by this step, all larger number within the array is already being pushed towards the end of the array accordingly. So the final array is 1,2,4,5,5,6,7,7,8,9,10,12,13,23

b. **write down the total number of key exchanges**: The total number of key exchanges is 12.

Question 3. Pseudocode for extract median: if we are to extract the median, the logic need to be that in an array of numbers, the median = (max + min)/2, since the median is the middle value:
Part 1 code:

```
Extract_Median(A)
        if A.heap() – size < 1 then
                error “heap underflorw”
        end if
        Median = A[1]
        A[1] = A[A.heap + size]
        A.heapSize  = A.heapSize/2
Extract_Median(A,1)
Return Median
```

Part 2:
The running time would be O(lg(n)), it is doing a constant amount of work on top of O(lg(n)) time.

for the questions from 4 to 6, I made table using % as mod symbol to get the reminder

Question 4:
Part 1:
When h(k) = k mod 9 , total collisions = 6, using mod 9 to get reminders

6	%	9	6
8	%	9	8
10	%	9	1
9	%	9	0
7	%	9	7
5	%	9	5
4	%	9	4
13	%	9	4
23	%	9	5
1	%	9	1
5	%	9	5
7	%	9	7
12	%	9	3
4	%	9	4

It will convert to, with index as first column and data on the 3rd column

0 ~ 9
1 ~ 10,1
2 ~
3 ~ 12
4 ~ 4,13,4
5 ~ 5,23,5
6 ~ 6
7 ~ 7,7
8 ~ 8

Part 2:
When h(k) = k mod 7, total collisions = 7, using mod 7 to get reminders

6 % 7 6
8 % 7 1
10 % 7 3
9 % 7 2
7 % 7 0
5 % 7 5
4 % 7 4
13 % 7 6
23 % 7 2
1 % 7 1
5 % 7 5
7 % 7 0
12 % 7 5
4 % 7 4

And it will convert to, with index as first column and data on the 3rd column

0 ~ 7,7
1 ~ 8,1
2 ~ 9,23
3 ~ 10
4 ~ 4,4
5 ~ 5,5,12
6 ~ 5,13

Question 5:

The number of collision of inserting these elements is 6.

31	%	7	3
11	%	7	4
5	%	7	5
17	%	7	3
25	%	7	4

It will convert to, with index as first column and data on the 3rd column

0	~	25
1	~	
2	~	
3	~	31
4	~	11
5	~	5
6	~	17

Question 6:

Using the formula h1(k) = k mod 7 and h2(k) = 1 + (k mod 3) to get the below steps.

31	%	7	3	3 plus 0	%	3	plus	0	equal to	3	%	7	3
11	%	7	4	1 plus 11	%	3	plus	3	equal to	7	%	7	0
5	%	7	5	1 plus 5	%	3	plus	6	equal to	11	%	7	4
17	%	7	4	1 plus 17	%	3	plus	6	equal to	12	%	7	5
25	%	7	3	1 plus 25	%	3	plus	8	equal to	12	%	7	5

And this converted to the following table, with index on the left, and data on the 3rd column

0	~	11
1	~	
2	~	
3	~	31
4	~	5
5	~	17,25
6	~	