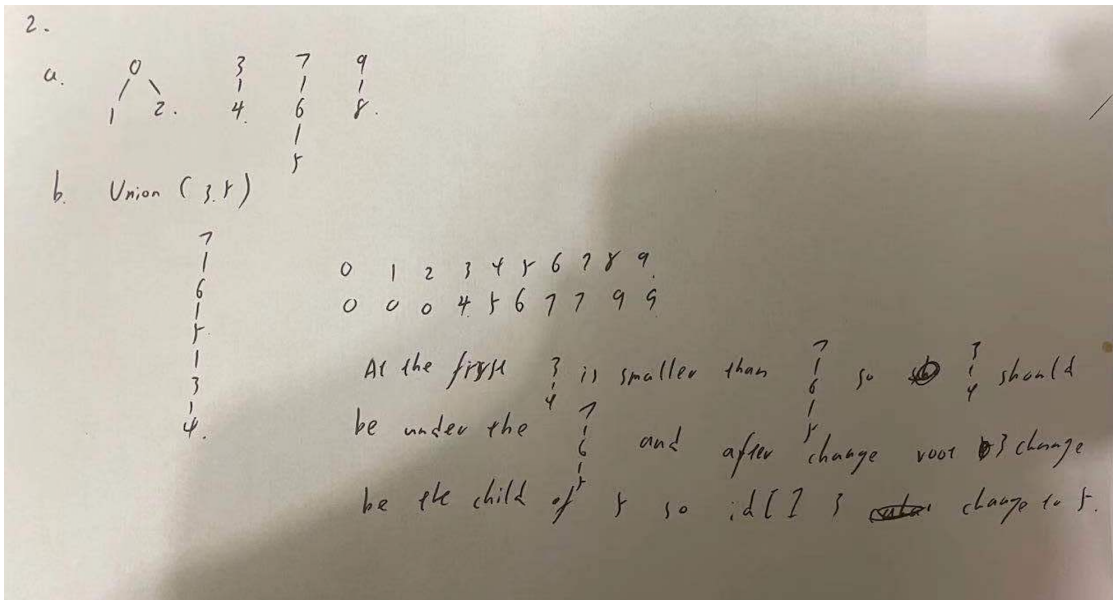
Name: Longyu Zhu
Student Number:400247195

1.b
2.d
3.b
4.c
5.b
6.c
7.c

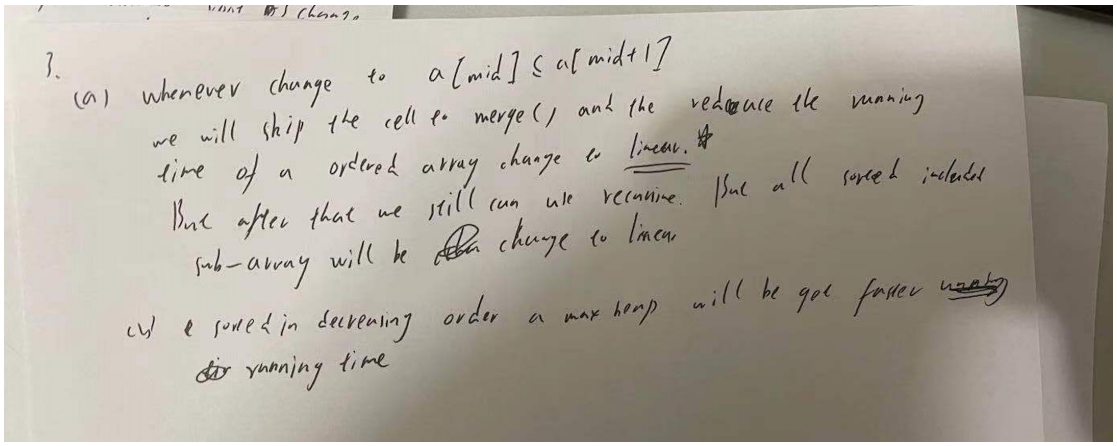
1.

```
procedure rotate_half (Node L, int which){//which meaning which one to be the center
if L.head==NUll return L.head;

if (which>L.size()){
int m;                                  //if which >L.size() that mean he need choose another one
scanner m;
rotate_half(L,m)
}

Node current =L.head; //[1 , 3 , 6 , 5 , 7 , 8]
for (int i=0;i<which,i++){
current=current.next
}

if (current.prev==NUll){ // when current.value=1
for (int i=0,i<L.size(),i++){
current = current.next;
}
Node left_Node=NUll;   //if the left side of current is null
                       // current =the end one list and make a new list to get a new list;
for (int i=0,i<L.size(),i++){
left_Node.next=current;
left_node=left_node.next;
current=current.prev;
}return Left_Node;  //after all put in to the left_node=[8 , 7 , 5 , 6 , 3 , 1]
}

if (current.next==NUll){
for (int i=0,i<L.size(),i++){
current = current.prev;
}
Node right_Node=NUll;
for (int i=0,i<L.size(),i++){
right_Node.next=current;          //if the center in the right side of L
right_node=right_node.next;
current=current.prev;
}return right_Node
}

//if current are not the first one or last one right now
Node L_buffer=current
NOde R_buffer=current;                      // use L and R make both side change
node buffer=current.head
for (int i =0;i< L.size()-which,i++){
buffer=buffer.next
}
for (int i =0;i< L.size()-which,i++){
R_buffer.next=buffer;
buffer=buffer.prev;
R_buffer=R_buffer.next
}
while (buffer.prev!=NUll){
buffer=buffer.prev;
}
for (int i =0;i< which-1,i++){
L_buffer.prev=buffer;
buffer=buffer.next;
L_buffer=L_buffer.prev
}
return L_buffer;             //because the L_buffer is the first one after .prev so return it.
}
```

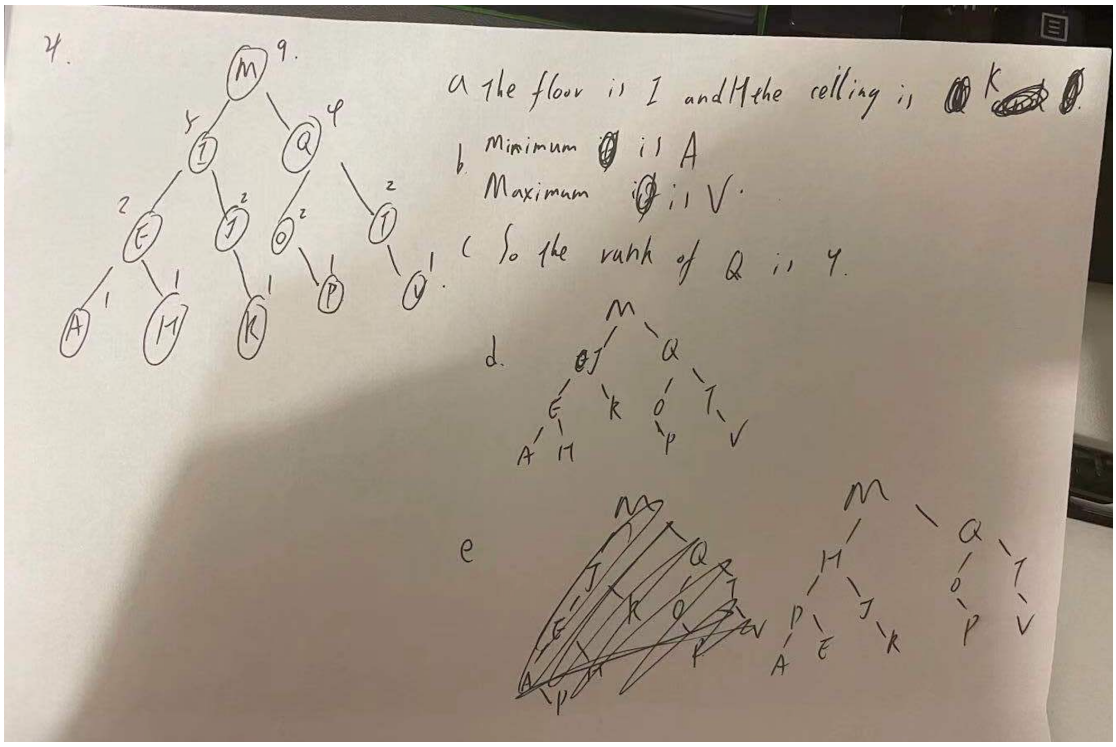
2.


2.
a.
b. Union (3,7)
0 1 2 3 4 5 6 7 8 9
0 0 0 4 5 6 7 7 9 9
At the first 3 is smaller than 7 so 3 should
be under the 7 and after change root of 3 change
be the child of 5 so id[3] 5 change to 5.

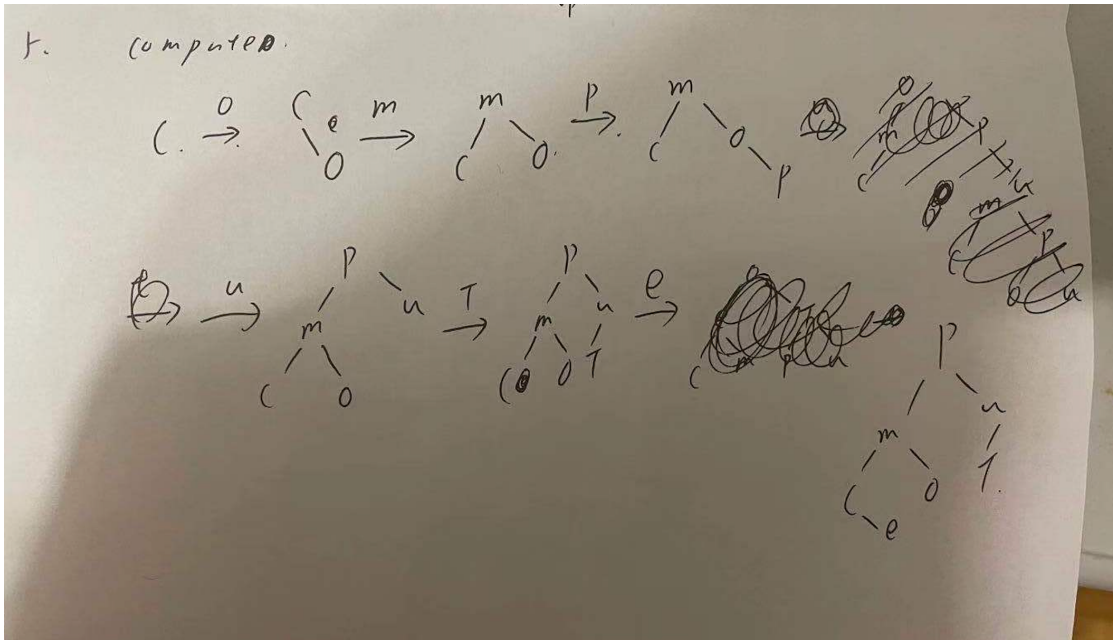
3


3.
(a) whenever change to a[mid] ≤ a[mid+1]
we will skip the cell to merge() and the reduce the running
time of a ordered array change to linear.
But after that we still can use recursive. But all sorted included
sub-array will be change to linear
(b) e sorted in decreasing order a max heap will be got faster
running time

4.


4.
a. the floor is I and M the ceiling is K.
b. Minimum is A
Maximum is V.
c. So the rank of Q is 4.
d.
e

5


5.
computed.