1)
```
int maximum () {
    TNode *bantu;
    int maks;
    if (isEmpty()) {
        return 0;
    }
    else {
        maks = head -> data;
        bantu = head -> next;
        while (bantu != NULL) {
            if ( bantu -> data > maks ) maks = bantu -> data
            bantu = bantu -> next
        }
        return maks
    }
```

2)

$A(1,4)$	$= A(0, A(1,3))$	kondisi 3
	$= A(0, A(0, A(1,2)))$	kondisi 3
	$= A(0, A(0, A(0, A(1,1))))$	kondisi 3
	$= A(0, A(0, A(0, A(0, A(1,0)))))$	kondisi 3
	$= A(0, A(0, A(0, A(0, A(0,1)))))$	kondisi 2
	$= A(0, A(0, A(0, A(0,2))))$	kondisi 1
	$= A(0, A(0, A(0,3)))$	kondisi 1
	$= A(0, A(0,4))$	kondisi 1
	$= A(0,5)$	kondisi 1
	$= 6$	kondisi 1

nilai akhir $A(1,4)$ adalah 6

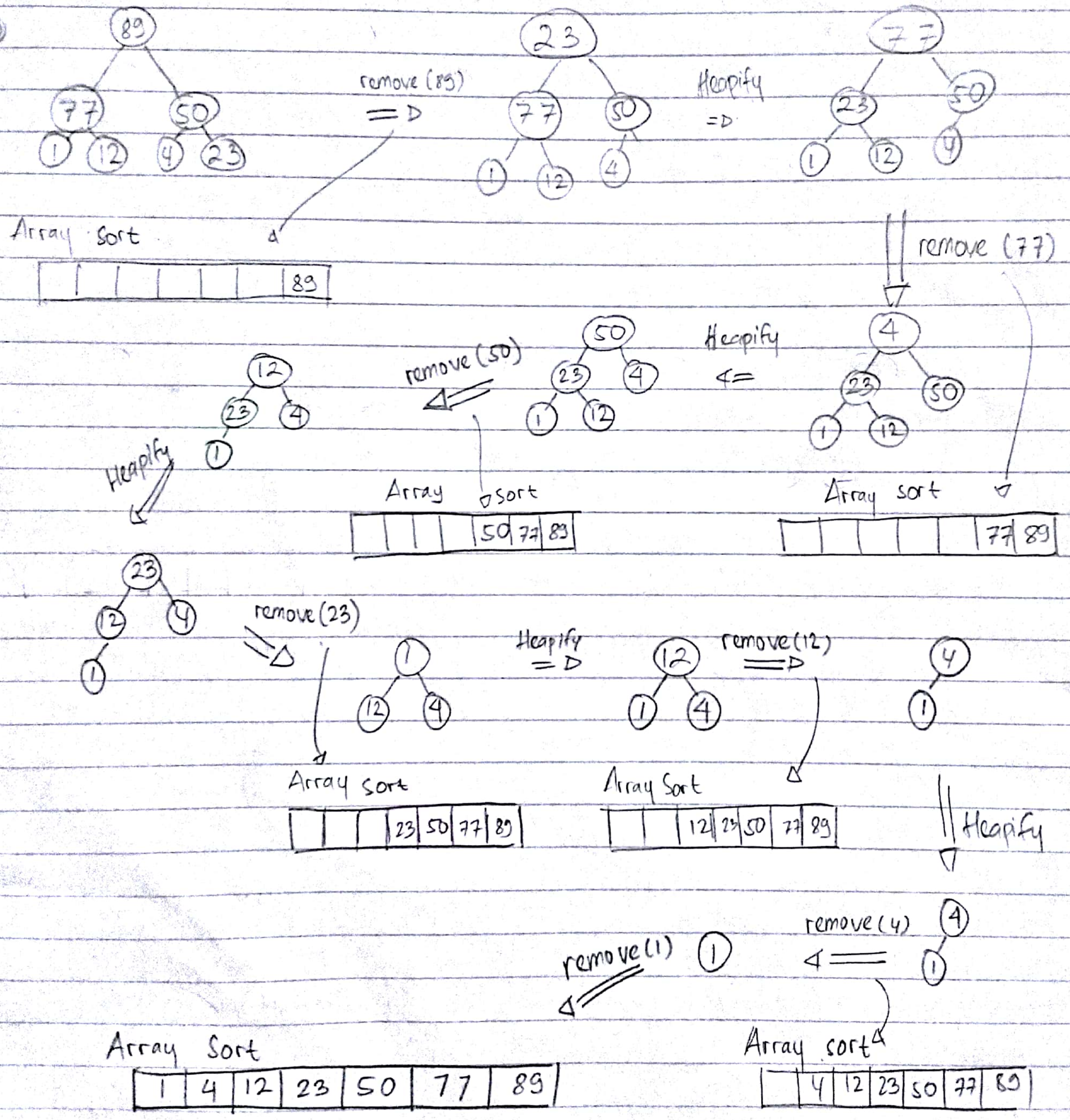
```
3)  typedef struct {
        int data;
        int height;
    } heightnode
    /* topViewPopulate prosedur untuk memberikan kordinat pada masing-masing Node */
    void topViewPopulate (struct node * root ,int h, int y ,heightnode * m[1000]){
        if (root == NULL) return ;

        int index = y % 1000 ;
        if (index < 0) index += 1000 ;

        if (m[index] == NULL or m[index]->height > h ){
            heightnode * hn = malloc (sizeof (heightnode)) ;
            hn -> data = root -> data ;
            hn -> height = h;
            m [index] = hn ;
        }
        topViewPopulate (root ->left , h+1 , y-1 , m) ;
        topViewPopulate( root->right, h+1, y+1, m) ;
    }

    Void topView (struct node * root){
        heightnode * m [1000] ;
        for (int i = 0 ; i< 1000 ; i++){
            m [i] = NULL;
        }
        topViewPopulate (root, 0, 0, m) ;
        for (int i = -500 ; i <500; i++){
            int index = i % 1000;
            if (index <0){
                index += 1000;
            }
            if (m [index] != NULL){
                printf (" %d  ", m [index] -> data );
            }
        }
        printf ("\n");
    }
```

4)


89
77
50
1
12
4
23
remove (89)
23
77
50
1
12
4
Heapify
77
23
50
1
12
4
Array sort
89
remove (77)
4
23
50
1
12
Heapify
50
23
4
1
12
Array sort
77 89
remove (50)
12
23
4
1
Array sort
50 77 89
Heapify
23
12
4
1
remove (23)
1
12
4
Heapify
12
1
4
remove(12)
4
1
Array sort
23 50 77 89
Array Sort
12 23 50 77 89
Heapify
4
1
remove (4)
1
remove(1)
Array sort
4 12 23 50 77 89
Array Sort
1 4 12 23 50 77 89