امیرحسین نجفی زاده 9831065

الف) بررسی کنید که آرایه زیر max-heap **هست یا خیر.اگر نیست** BUILD-MAX-HEAP **را روی آن اجرا کنید و مراحل اجرای آن را نشان دهید.**

21,15,17,25,23,30,24,28,13,12,20,19,10

بدیهی است که اعداد فوق max-heap **تشکیل نمی دهند چون باید آرایه با ارزش** 30 **در اول آرایه حضور می داشت.**

برای درست کردن یک max-heap **روی اعضای** 1 **تا** n / 2 **تابع** max-heapify **را صدا می زنیم.**

تعداد اعضای ما برابر 13 **تا است. پس ایندکس های** 1 **تا** 6 **را** max-heapify **می کنیم. (به صورت نزولی از** 6 **به** 1 **)**

Max-heapify(A[6]) :

1. A[6] => 30
2. L = left [6] = A[12] = 19 , R = right [6] = A[13] = 10
3. Largest (30, 19, 10) = 30 => no change

Max-heapify(A[5]):

1. A[5] => 23
2. L = left [5] = A[10] = 12 , R = right [5] = A[11] = 20
3. Largest (23, 12, 20) = 23 => no change

Max-heapfiy(A[4]):

1. A[4] = 25
2. L = left[4] = A[8] = 28 , R = right[4] = A[9] = 13
3. L > A[4] => A[4] = 28 , A[8] = 25
4. Max-heapify(A[8]) => 8 is leaf

Max-heapify(A[3]):

1. A[3] = 17
2. L = left[3] = A[6] = 30 , R = right[3] = A[7] = 24
3. L > A[3] => A[3] = 30 , A[6] = 17
4. Max-heapify(A[6])
    1. A[6] = 17
    2. L = A[12] = 19 , R = A[13] = 10
    3. L > A[6] => A[6] = 19 , A[12] = 17
    4. Max-heapify(A[12]) => 12 is leaf

Max-heapify(A[2]):

1. A[2] = 15
2. L = left[2] = A[4] = 28 , R = right[2] = A[5] = 23
3. L > A[2] => A[2] = 28, A[4] = 15

4. Max-heapify(A[4])
    1. A[4] = 15
    2. L = A[8] = 25 , R = A[9] = 13
    3. L > A[4] => A[4] = 25 , A[8] = 18
    4. Max-heapify(A[8]) => 8 is leaf

Max-heapfiy(A[1]):

1. A[1] = 21
2. L = left[1] = A[2] = 28 , R = right[1] = A[3] = 30
3. L > A[1] => largest = 2
4. R > A[largest] => A[3] = 21 , A[1] = 30
5. Max-heapfiy(A[3]):
    1. A[3] = 21
    2. L = A[6] = 19, R = A[7] = 24
    3. R > A[3] => A[3] = 24 , A[7] = 21
    4. Max-heapfiy(A[7]) => 7 is leaf

And we are done, array is :

30 | 28 | 24 | 25 | 23 | 19 | 21 | 15 | 13 | 12 | 20 | 17 | 10

ب) به ازای فراخوانی های زیر ، درخت نهایی را رسم کنید.(فراخوانی ها را به ترتیب قرار گرفته شده انجام دهید.)

1. `HEAP-EXTRACT-MAX(A)`
2. `HEAP-INCREASE-KEY(A ,i=4 ,key=40)` **منظور از** `i`**، خانه ای از آرایه که باید به مقدار** K **افزایش یابد.**
3. `MAX-HEAP-INSERT(A ,key=38 )` **منظور از** `k` **مقداری است که باید وارد هیپ شود**

1. Extract max (A)
    1. Get a copy of 30
    2. Replace 30 with 10 => remove 10 index which is 13 form array
    3. Max-heapify(A[1]):
        1. A[1] = 10
        2. L = A[2] = 28 , R = A[3] = 24
        3. L > A[1] => A[1] = 28 , A[2] = 10
        4. Max-heapify(A[2]):
            1. A[2] = 10
            2. L = A[4] = 25 , R = A[5] = 23
            3. L > A[2] => A[2] = 25 , A[4] = 10
            4. Max-heapfiy(A[4]):
                1. A[4] = 10
                2. L = A[8] = 15 , R = A[9] = 13
                3. L > A[4] => A[4] = 15 , A[8] = 10

4. Max-heapify(A[8]) => 8 is leaf

Array is : 28 | 25 | 24 | 15 | 23 | 19 | 21 | 10 | 13 | 12 | 20 | 17

2. Heap increase key = 40 for I = 4
    1. A[4] = 15
    2. A[4] = 40
    3. P[4] = 2 => A[2] < A[4] => A[2] = 40 , A[4] = 25
    4. P[2] = 1 => A[1] < A[2] => A[1] = 40 , A[2] = 28

Array is : 40 | 28 | 24 | 25 | 23 | 19 | 21 | 10 | 13 | 12 | 20 | 17

3. Max-heap insert 38
    1. A[13] = 38
    2. P[13] = A[6]
    3. A[6] = 19 < A[13] => A[13] = 19 , A[6] = 38
    4. P[6] = A[3]
    5. A[3] = 24 < A[6] => A[6] = 24 , A[3] = 38
    6. P[3] = A[1]
    7. A[1] > A[3] => done

Array is : 40 | 28 | 38 | 25 | 23 | 24 | 21 | 10 | 13 | 12 | 20 | 17 | 19