HW1

① show Complexity of this program (array is sorted).
(explain your answer)

```
int BinarySearch (int array[], int n, int value){
int left=1, right=n, middle;
while (right >= left){
    middle =(left +right)/2;
    if (array [middle] == value)
        return middle;
    if (array [middle] > value)
        right = middle;
    else
        left = middle;
}
return -1;
}
```

فرض می‌کنیم طول آرایه N باشد و عنصر مورد نظر در مرحله k ام پیدا شود. طبق برنامه بالا، در هر مرحله، طول آرایه نصف می‌شود. بنابراین داریم:

$$N \rightarrow \frac{N}{2} \rightarrow \frac{\frac{N}{2}}{2} \rightarrow \overset{\text{مرحله } k \text{ ام}}{\cdots} \rightarrow \frac{N}{2^k}$$

علاوه بر این می‌دانیم زمانی که عنصر مورد نظر پیدا می‌شود، طول آرایه 1 است و آن عنصر هم

عنصر مورد نظرایت، بنابراین داریم:

$$L = \frac{N}{2^k}$$

$$\text{if } L=1 : \quad 1 = \frac{N}{2^k} \rightarrow 2^k = N \rightarrow k = \log_2^N$$

② Calculate $T(n)$ runtime of these recursion functions with master method (explain your answer)

a) $T(n) = 3T(\frac{n}{2}) + n$

$a=3$, $b=2$, $f(n)=n$, $n^{\log_b^a} = n^{\log_2^3} \simeq n^{1.58}$

$f(n) = O(n^{\log_2^3}) \rightarrow T(n) = \Theta(n^{\log_2^3})$ Case 1

b) $T(n) = T(\frac{n}{2}) + 2^n$

$a=1$, $b=2$, $f(n)=2^n$, $n^{\log_b^a} = n^{\log_2^1} = 1$

$2^n > 1 \rightarrow T(n) = \Theta(2^n)$ Case 3

c) $T(n) = 3T(\frac{n}{2}) + n^2$

$a=3$, $b=2$, $f(n)=n^2$, $n^{\log_b^a} = n^{\log_2^3} \simeq n^{1.58}$

$n^{1.58} < n^2 \rightarrow T(n) = \Theta(n^2)$ Case 3

d) $T(n) = 8T(\frac{n}{2}) + n^3$

$a = 8$, $b = 2$, $f(n) = n^3$, $n^{\log_b^a} = n^{\log_2^8} = n^3$

$n^3 = n^3 \longrightarrow T(n) = \theta(n^3 \log^n)$ Case 2

③ Apply insertion Sort to the following arrays and write all steps:

a) [72, 34, 2, -5, 13, -6, 4, 3]

① [12, 34, 2, -5, 13, -6, 4, 3] ② [2, 12, 34, -5, 13, -6, 4, 3]

③ [-5, 2, 12, 34, 13, -6, 4, 3] ④ [-5, 2, 12, 13, 34, -6, 4, 3]

⑤ [-6, -5, 2, 12, 13, 34, 4, 3] ⑥ [-6, -5, 2, 4, 12, 13, 34, 3]

⑦ [-6, -5, 2, 3, 4, 12, 13, 34]

b) [8, 3, 23, -4, 16, 22]

① [3, 8, 23, -4, 16, 22] ② [3, 8, 23, -4, 16, 22]

③ [-4, 3, 8, 23, 16, 22] ④ [-4, 3, 8, 16, 23, 22]

⑤ [-4, 3, 8, 16, 22, 23]

c) [9, -8, 4, 56, 23, 11, 6, 42]

① [-8, 9, 4, 56, 23, 11, 6, 42] ② [-8, 4, 9, 56, 23, 11, 6, 42]

③ [-8, 4, 9, 56, 23, 11, 6, 42] ④ [-8, 4, 9, 23, 56, 11, 6, 42]

⑤ [-8, 4, 9, 11, 23, 56, 6, 42] ⑥ [-8, 4, 6, 9, 11, 23, 56, 42]

⑦ [-8, 4, 6, 9, 11, 23, 42, 56]

④ Suppose we have an array, [1, 3, 6, 7, 10], and a new element to add, x = 4. If we use the insertion sort to add x to our array, how many comparisons will it take and what will be the final index of x in the array

Solution 4: [1, 3, 6, 7, 10, 4]

① [1, 3, 6, 7, 10, 4]

② [1, 3, 6, 7, 10, 4]

③ [1, 3, 6, 7, 10, 4]

④ [1, 3, 6, 7, 10, 4]

⑤ [1, 3, 4, 6, 7, 10]

↓

index = 2

(5) Consider the program below and then answer the questions posed:

a) Implement the insertFirst method that inserts a new node with the given data key at the front of the linked list. Assume the list is not empty.

b) Implement the delete method that deletes a node with the given data key from the related LinkList object, and returns the Node containing the key. Assume the list is not empty. If the key does not exist in the list, null should be returned and no action should be taken.

c) Predict the printed results of the main method in LinkListApp.

```
class Node {
   public int iData;  // key
   public Node next;
   public Node (int id)
   {
```

(5)

```
first = null;
{

public Nale find (int key)
{
    Node current = first;
    while (current != null && current.iData != key)
        current = current.next;
    return current
}

public void displayList()
{
    for (Node current = first; current != null; current = current.next)
        System.out.println (current.iData);
}
```

(5)

```
Public   void  insertFirst (int key)
{
    questiona
}

Public  Nale  delete (int key)
{

    question b
}
}

class  LinkLst App
{

public  static  void  main (String [] args)
{

    LinkList  theList = new  LinkList ();
     the List. insertFirst (22) ;
     the List. insert First (44);
     theList. insertEirst (66);
    Node d= the List. delete (44);
    d = theList .. delete (88),
    theList. displayList ();
}
}
```

a)

```
Node node = first;
first = new Node (key);
first.next = node;
```

b) {

```
Node n = first;
Node del = find (key);
while( n.next.iData != key){
        n = n.next;
}
if ( del != null) {
        n.next = del.next;
}
return del;
}
```

c) 66 22