**DS: Home-Task 2**

**Q1:**

**a)**

**First loop = n/2 times (**for first execution later n times)

**2nd loop = log2n**

* **O(nlog(n))** times

**b)**

**log2n**

* **O(logn)**

**c)**

**First loop = n/2 times (**for first execution later n times)

**Second and third = log2n**

* **O(n(logn)2)**

**d)**

* **O(log(log(n)))**

**Q2)**

* + - Input **xpos** and **ypos** from the User;
    - Create a **variable** and store the **xpos index** value.
    - Assign **Xpos** index value to **Y Index** Value;
    - Lastly, Assign **temp value** to y index.

int temp = A[xpos];

A[xpos] = A[ypos];

A[ypos] = temp;

**------To check replacing number is increasing or decreasing-----**

Firstly we check if **xpos is at 0 index** and replace with any **y index value** if its true then if value at xpos Position is less than y position then a number would be **increasing order** else it would be decreasing.

If xpos is at any other index other than 0 then we simply compare the value at xpos with y index value, if the value at x index is less than y index then number would be in **increasing order** else the number would be **decreasing order. great**

**Special Case:** if y index is greater than x index then make x index as y and y as x.

**Q3)**

**To check that** is their any cycle exist in the linked list;

Use two pointers **Current and Previous**

Apply a condition if Head of the linked list is null simply **return false.**

**Previous pointer** would be pointing to Head and Current Pointer would be pointing to **head’s next**

Make a loop to till the Previous not equal to Current

Wrote condition than if **Current is equal** to Null return false

**Else if Current’s next is equal to NULL** again return false

And make increment Current pointer by 2 steps and Previous by 1 step

Loop will be proceeded until Previous became equal to Current

If return false conditon not found in this case or simply Null not found then it would be a cyclic.

**Q4)**

Make a **pointer temp** which is pointing to Head and create a **counter** variable intializing to zero and boolean flag which is equal to false.

Make a loop till **temp** not equal to NULL

Make counter++ in the loop and write condition in the loop if **temp** **->data > element** then set **flag to true**. It means we have find our index.then **break loop.**

Check if **flag false** then It means element would be at the end because we can’t find greater value because moand print input is not valid and return.

**If its true**

If Head is equal to Null then Make a new Node **ElementNode** on heap

assign **ElementNode** **->data = element**

**ElementNode** ->prev=**NULL**;

**ElementNode** ->next=**Head**;

**Head**->prev= **ElementNode**;

**Head**= **ElementNode**;

else If index is greater than 0

Than start a loop till counter-1 times

And increment a pointer **temp2** which is pointing to **Head**

Then pointers would be reach at appropriate position and

then make a newNode **ElementNode.**

**ElementNode->data = Element.**

**ElementNode ->prev=temp2;**

**ElementNode ->next= temp2->next;**

**if(temp2->next)**

**temp2->next->prev= ElementNode;**

**temp2->next= ElementNode;**

**The time complexity would be o(logn) when elements are in sorted and binary search operation performed.**

**Q5)**

1. The linked list would be the most appropriate because we can’t use array it has fixed size issue and insertion in linked list is most efficient as compared to Array because for array we need to shift the whole array.
2. Start with the pointer **temp2** pointing to head and make a loop if **temp2**!=null or **temp2 ->data>Item**.

Then make a condition if pointer’s data equal to key then return the node or show the information else no id found and incrementing the loop.

1. If(Head==NULL)

Then make a newNode **EleNode**

**EleNode** -> Next = **Head**;

**Head** = **EleNode** and **EleNode**

Else

make a pointer **temp2** pointing to Head and start a loop till NULL and increment the **counter++** by comparing ID’s of the Employee if ID became Greater **Break**

And then going to appriopriate place by using loop till **counter-1** make a new node **EleNode** and **EleNode**->**next** is equal to **temp2**->**next** and **temp2->next** assign to **EleNode**.

**Q6)**

The data structure that we would use is **Stack** because in stack last element pushed would come out first.

Then we push elements in Stack

Create a newNode NodeEle with given value.

Make a condition **if(top==NULL)**

Then **NodeEle->next = NULL**

else

**newEle->next = top**

**top = newEle**

**Then Elements would be printing reverse order as the property of stack last in first out so elements would be print in reverse Order.**