Muhammad Faizan (13205)

Question (1) (1)

Q(1) (1) Explain the following operations in a single link list.

(1) Insert an Element (2) Delete an Element

Insertion:

In a single linked list, the insertion operation can be performed in three ways. They are as follows:-

(1) Inserting At Beginning of the list
(2) Inserting At end of the list
(3) Inserting at specific location in the list

Inserting at Beginning of the list:

We can use following steps to insert a new node at Beginning of the single linked list:-

⇒ Step 1 - Create a newNode with given value
⇒ Step - 2 Check whether list is Empty (head == NULL)
⇒ Step - 3 If it is Empty then, set newNode → next = NULL and head = newNode
⇒ Step 4 - If it is not Empty then, set newNode → next = head and head = newNode

Inserting at End of list:-

We can use following steps to insert a new node at end of single linked list.

⇒ Step 1 - Create a newNode with given value and newNode → next as NULL
⇒ Step 2 - Check whether list in Empty (head == NULL)
⇒ Step 3 - If it is Empty then, set head = newNode
⇒ Step 4 - If it is Not Empty then, define a node pointer temp and initialize with head
⇒ Step 5 - Keep moving the temp to its next

node until it reaches to the last node in the list (until temp → next is equal to NULL).
⇒ Step 6 - set temp → next = newNode.

Inserting At specific location (After a Node):-

We can use following steps to insert a new node after a node in the single linked list.

⇒ Step 1 - Create a newNode with given value
⇒ Step 2 - Check wheather list in Empty (head == NULL)
⇒ Step 3 - If it is Empty then, set newNode → next = NULL and head = newNode.
⇒ Step 4 - If it is Not Empty then, define a node pointer temp and initialize with head.
⇒ Step 5 - Keep moving the temp to its next node until it reaches to the node after which we want to insert the newNode (until temp 1 → data is equal to location, here location is the node value after which we want to insert the newNode).
⇒ Step 6 - Every time check wheather temp is reached to last node or not. If it is reached to last node then display "Given node is not found in the List! Insertion not possible! and terminate the function. Otherwise move the temp to next node.
⇒ Step 7 - Finally set newNode → next = temp → next and temp → next = newNode.

(2) Deletion :-

In a single linked list, the deletion operation can be performed in three ways. They are as follows:

(1) Deleting from Beginning of the list
(2) Deleting from End of the list
(3) Deleting a Specific Node

* Deleting from Beginning of the list :-

We can use the following steps to delete a node from beginning of the single linked list.

⇒ Step 1 - Check whether list is Empty (head == NULL)
⇒ Step 2 - If it is Empty then display "List is Empty! Deletion is not possible" and terminate the function.
⇒ Step 3 - If it is Not Empty then, define a Node pointer temp and initialize with head.
⇒ Step 4 - Check whether list is having only one node (temp -> next == NULL)
⇒ Step 5 - If it is True then set head = NULL and delete temp (Setting Empty list conditions)
⇒ Step 6 - If it is False then set head = temp -> next and delete temp.

* Deleting from End of the list :-

We can use following steps to delete a node from end of the single linked list.

⇒ Step 1 - Check whether list is Empty (head == NULL)
⇒ Step 2 - If it is Empty then display "List is Empty!!! Deletion is not possible and terminate the function.

⇒ Step 3 - If it is Not Empty then define two Node pointer temp 1 & temp 2 and initialize temp1 with head.
⇒ Step 4 - Check whether list has only one Node (temp 1 -> next == NULL)
⇒ Step 5 - If it is True then set head = NULL and delete temp 1 and terminate the function
⇒ Step 6 - If it is False then set temp 2 = temp 1 & move temp 1 to its next node.
(temp 1 -> next == NULL)
⇒ Step 7 - Finally set temp 2 -> next = NULL & delete temp 1.

Deleting a Specific Node From List :-

We can use following step to delete specific node from single linked list.
⇒ Step 1 - Check whether list is Empty (head == NULL)
⇒ Step 2 - If it is Empty then display list is Empty! Deletion is not possible and terminate the function.
⇒ Step 3 - If it is Not Empty then define two Node pointers temp 1 & temp 2 & initialize temp1 with head.
⇒ Step 4 - Keep moving the temp 1, and every time set temp 2 = temp 1 before moving the temp 1 to its next node.
⇒ Step 5 - If it reached to last node then display Given node not found in list Deletion not possible.

M. Faizan (13205)

Question (2) (5)

① ② Write a program to insert a new element in given unsorted array at k^{th} position.

Ans **Program to Insert New element :-**

```
#include<stdio.h>

 void main()
 {
  int arr1[100],i,k,p,x;
    printf("\n\kInsert New value in the unsorted
array :\n");
    printf("_____\n");
    printf("Input the size of array:");
    scanf("%d",&k);
  /* unsorted values into the array*/
    printf("Input %d elements in the array in
ascending order:\n",n);
    for(i=0;i<k;i++)
       {
        printf("element-%d:",i);
        scanf("%d",&arr1[i]);
       }
 printf("Input the value to be inserted:");
  scanf("%d",&x);
 printf("Input the Position, where the value
 to be inserted:");
  scanf("%d",&p);

  printf("The current list of the array:\n");
   for(i=0;i<k;i++)
    printf("%5d",arr1[i]);
```

```
/*Move all data at right side of the array
*/
  for(i=k;i>=p;i--)
    arr1[i]=arr1[i-1];
  /* insert value at given position*/
    arr1[p-1]=x;

  printf("\n\nAfter Insert the element the new
    list is :\n");
    for(i=0;i<=k;i++)
    printf("%5d", arr1[i]);
    printf("\n\n");
  }
```

Question (3)

(3) Explain Quick sort with help of suitable example.

Ans **Quick Sort** ..

It is sorting algorithm which is commonly used in Computer Science. Quick Sort is a divide and conquer algorithm. It creates two empty array to hold element less then the pivot value and element greater then the pivot value and then recursively sort the sub array. There are two basic operations in the algorithm swapping items in place and partitioning a section of the array.

Example ..

```
arry=[9,2,5,6,4,3,7,10,1,12,8,11];
function quickSort(array)
if(array.length==0) return[];
var left=[],right=[],pivot=array[0];
for(var i=1;i<array.length;i++){
```

```
if (array[i] < pivot)
    left.push(array[i])
Else
    right.push(array[i]);
}
return quickSort(left).concat(pivot, quick
    Sort(right));
}
Console.log(quickSort(array.slice())); //->
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12].
```

Q 3(b) Write an algorithm for insertion in sorted linked list.

Ans: **Algorithm for insertion in sorted linked list :**

⇒ Step 1 :- Determine pivot

4	2	6	5	3	9

⇒ Step 2 :- Start pointers at left & right

4	2	6	5	3	9

⇒ Step 3 :- Since 4 < 5, shift left pointer

4	2	6	5	3	9

⇒ Step 4 :- Since 2 < 5, shift left pointer
Since 6 > 5, stop

M. Faizan (13205)

(8)

4	2	6	5	3	9

⇒ Step 5

Since 9 > 5 shift right pointer
Since 3 > 5 ; Stop

4	2	6	5	3	9

⇒ Step 6

Swap values at pointers

4	2	3	5	6	9

Step 7

~~Swap~~ move pointer one more step

4 2 3 5 6 9

Step 8

Since 5 == 5
move pointer one more step
Stop

4 2 3 5 6 9

—o

M. Faizan (13205)

Question (4) (9)

Q(a) The adjacency list representation of a graph with five vertices A, B, C, D, E is given below.

Part (a) Draw the adjacency matrix :-

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	0
C	0	1	0	1	1
D	0	1	1	0	0
E	1	0	1	0	0

Part (b) Draw the graph :-