Name:- UMAR KHAN

ID :- 13079

Date :- 25 sep 2020

Subject:- Data Structure

Final term

Exam

Q1. Explain the following operation in single link a list!

1 Insert an element
2 Delete an element.

Ans

Single link list :-

Simply a list is a Sequence of data, and the linked list is a sequence of data linked with eacto other.
The formal definition of a single linked list is as follows.--
Single linked list is a sequence of elements in which every element has link to its next element in the sequence.

Insertion :-

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

1. Inserting At Beginig of the list
2. Inserting At End of the list

3. Inserting At specific location in the list

Inserting At Begining of the 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 the list

step 1 - Create a newNode with given value and newNode -> next as Null.

step 2 - Check whether list is 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 initialized 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 in the list (After a Node)

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, define a node pointer temp and initialized with head.

Step5-

Keep moving the temp to its next node until it reaches to the node after which we want to insert the newNode (until temp1 -> data is equal to location, here location is the node value After which we want to insert the newNode;

Step6.

Every time check whether temp is 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'

Deletion :-

1. Deleting from Begining of the list
2. Deleting from End of the list
3. Deleting a specific Node.

Deleting From Begining of the List

Step1 -

Check wether list is empty (head == NULL)

Step2 -

If it is empty then, display "List is Empty!!! Deletion is not Possible' and terminate the function.

Step3 -

If it is not Empty then, define a Node pointer 'temp' and initialized with head.

Step4 -

Check wether list is having only one node (temp->next==N

Step5 -

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

Step 1-

Check wether 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 'temp1' and 'temp2' and initialized 'temp1' with head.

Step 4-

Check wether list has only one Node (temp1 → next == NULL)

Step 5-

If it is TRUE. Then set head = NULL and delete temp1. And terminate the function. (Setting Empty List Condition)

Step 6-

If it is false. Then, set 'temp2 = temp1' and move temp1 to its next node. Repeat the same until it reaches to the node in the ~~list~~ list. (until temp1 → next == NULL)

Step 7-

Finally, Set temp2 → next = NULL and delete temp1.

Q2

Ans: Coding

```
#include < stdio.h>

Void main ()
{
   int arr1[100], i, k, P, x;

   Printf("\n\kInseet New value
   inthe unSorted array:\n");
   Printf(" ----------\n");
   Printf("input the size of
   array : ");
   Scanf("%d", &k);
/* unStored Values into array*/
Printf("Input %d element in
the array in assending
order:\n", n);
for (i=0; i<k; i++)
{

   Pritf("element - %d; ", i);
   Scanf("%d", and arr1[i]);

}
```

```
Printf("Input the value to be
 insted:");
 Scanf("%d", and x);
printf("Input the Position,
 where the value to be
 Inserted:");
 Sanf("%d", and 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\n after Insert the
 element the new list
  is :\n");
 for(i=0;i<=k;i++)
  printf("%5d", arr1[i]);
    printf("\n\n");

}.
```

Q3 Explain Quick sort with the help of suitable example

Write an algorithm for insertion in sorted linked list?

Answer:-

Quick sort:-

Like Merge Sort, Quick Sort is a divide and conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quick sort that pick pivot in different ways.

1. Always pick first element as pivot.
2. Always pick last element as pivot.
3. Pick a random element as pivot.
4. Pick median as pivot.

For example:-

In the array {52, 37, 63, 14, 17, 8, 6, 25} we take 25 as Pivot so after the first Pass the list will be Changed like this.

{6 8 17 14 25 63 37-52}.

Hence after Pass, Pivot will be ~~Changed~~ set at its position, with all the elements Smaller to it on its left and all the elements larger then to its right.

Algorithm:-

step 1:- If it is the first element. it is already sorted. return 1;

step 2:- Pick next element.

Step 3= Compare with all elements in the sorted sub-list.

step 4= Shift all the element in the sorted sub-list that is greater than the value to be sorted.

step

steps - Insert the value.

Step - Repeat until list is sorted.

Insert in a sorted list

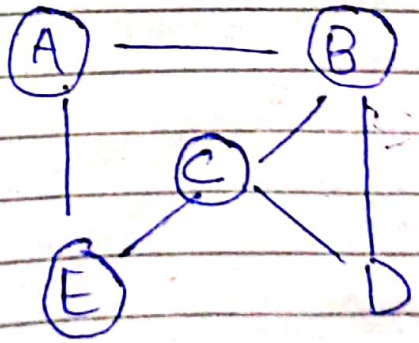
```
#include <bits/stdc++.h>
using namespace std;

/* Link list node */
class Node {
public:
    int data;
    Node* next;
};
```

Q 4:-

Adjacent 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	0	1	0	0
E	1	0	1	0	0

Graph:-



This is the graphic representation.