Exercise 1 Linked List

1. Write a C ++ Program for implementation of Linked List performing following operations
a) Insertion b) Deletion

Objective: The objective of this exercise is to enable you to program a Singly Linked List and to perform operations on it.

Procedure and description:
Linked list is a linear collection of data elements called nodes. Each node is divided into two parts: the first part is called data field where the data are stored and the second part is called the link field or next field, contains the address of the next node in the list.

a) Insertion:
An element can be inserted into linked list in two possible ways. The first one is inserting a node at the beginning of a list and the second one is inserting after the node with a given location.

i) Inserting at the beginning of a list
The simplest way of inserting a node is by inserting a node at the beginning of a list. So the algorithm is as follows.

Notations Used:
DATA : Node First part called DATA, contains data value
LINK : Node Second part called LINK, contains address
START : The address of the first node
AVAIL : Address of the pointer in the linked list
ITEM : New data (Node) to be added to the list
LOC : Address, where the New data (Node) is to be inserted

Definition of node:

```
Class Node
Start
    INT     DATA
    Node   *LINK
End
```

Algorithm: Inserting at the beginning of list
INSERT (DATA, LINK, START, AVAIL, ITEM)
Step 1: [OVERFLOW?]
If AVAIL = NULL, then: Write: OVERFLOW, and Exit.
Step 2: [Remove first node from AVAIL list]
Set N: = AVAIL and AVAIL: = LINK [AVAIL].
Step 3: Set DATA [N]:= ITEM. [Copies new data into new node]
Step 4: Set LINK [N]:= START. [New node points to the original first node]
Step 5: Set START: = N. [changes START so it points to the new node]
Step 6: Exit.

Executing insertion operation one must check for the OVERFLOW condition, i.e. check whether there is room for the new item in the list.

ii) Inserting after a given node

In this method we will give the location LOC of a node or the location will be null (LOC = NULL). So, here if the location of some node is given we insert the ITEM next to that node and if location is null then the ITEM will be the first node. The following is the algorithm to insert after a given node.

Algorithm:
INSERT1 (DATA, LINK, START, AVAIL, LOC, ITEM)
Step 1: [OVERFLOW?]
If AVAIL = NULL, then: Write: OVERFLOW, and Exit
Step 2: [Remove first node from AVAIL list]
Set N: = AVAIL and AVAIL: = LINK [AVAIL].
Step 3: Set DATA [N]:= ITEM. [Copies new data into new node]
Step 4: If LOC = NULL, then: [Insert as first node]
Set LINK [N]:= START and START: = N.
Else: [Insert after node with location LOC]
Set LINK [N]:= LINK [LOC] and LINK [LOC]:= N.
[End of if structure.]
Step 5: Exit

Expected Output:

After executing this program. Enter input choice as insertion operation.

i) Inserting at beginning of node:

For inserting element at beginning “enter choice insertion at beginning” .For better understanding see below pictorial representation.

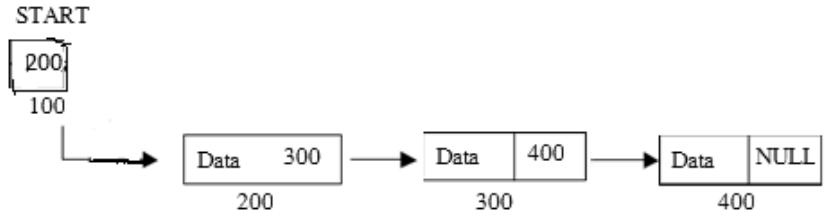

Fig.: Initial Linked List

From above pictorial representation

100 → The address of the first node (In single linked list first node Contains only next node address there is no data field)

200 → Then next node address

Fig.: After inserting node at beginning (node address: 500)

ii) Inserting element after given node

For inserting element for a given node. First enter node value and then enter data (element value). For better understanding see below pictorial representation.

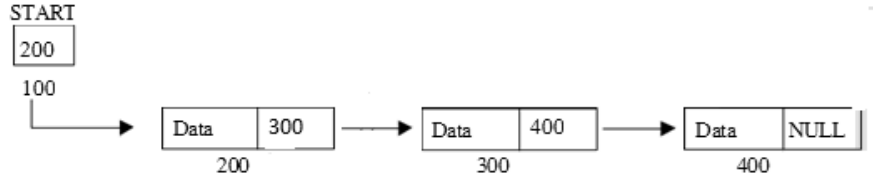

Fig.: Initial Linked List

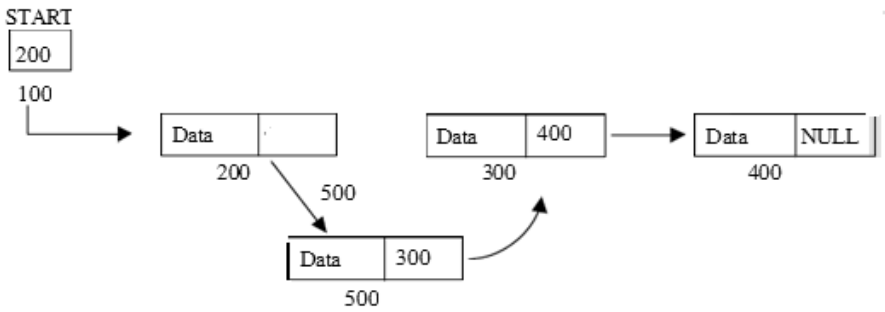

Fig.: After inserting node at given location (Example: LOC address 200)

b) Deletion

Deletion is the process of removing a node from the linked list. The deletion of a node is made just by a pointer change.

i) Deleting the node at beginning of list

Let N be the node to be deleted. The following algorithm deletes N from the list.

Algorithm:

DELETE (DATA, LINK, START, AVAIL, LOC, LOC1)
Step 1: If LOC1 = NULL, then:
Set START: = LINK [START]. [Deletes first node]
Else:
Set LINK [LOC1]:= LINK [LOC]. [Deletes node N]
[End of If structure]
Step 2: [Return deleted node to the AVAIL list]
Set LINK [LOC]:= AVAIL and AVAIL: = LOC.
Step 3: Exit.

ii) Deleting the node with a given ITEM of information

In this deletion method the ITEM information will be given and we have to delete the first node N which contains the ITEM. So, to delete the node N from the list we need to know the location of the node preceding N.

Algorithm:

FIND (DATA, LINK, START, ITEM, LOC, LOC1)
Find the location LOC and LOC1 of the node N and the node preceding N respectively.

Step 1: [List empty?] if START = NULL, then:
Set LOC: = NULL and LOC1:= NULL, and Return.
[End of If structure.]

Step 2: [ITEM in first node?] If DATA [START] = ITEM, them:
Set LOC: = START and LOC1:= NULL, and Return.
[End of If structure.]

Step 3: Set P1:= START and P: = LINK [START].
[Initializes pointers.]

Step 4: Repeat Steps 5 and 6 while P ≠ NULL.

Step 5: If DATA [P] = ITEM, then:
Set LOC: = P and LOC1:= P1, and Return.
[End of If structure.]

Step 6: Set P1:= P and P: = LINK [P]. [Updates pointers]
[End of Step 4 loop]

Step 7: Set LOC: = NULL. [Search Unsuccessful]

Step 8: Return.

Expected Output:
After executing this program. Enter input choice as deletion operation.

i) Deletion of element at beginning of list
For deleting element at beginning. Enter choice as deleting element at beginning. See below pictorial representation for better understanding.

Fig.: Initial Linked List

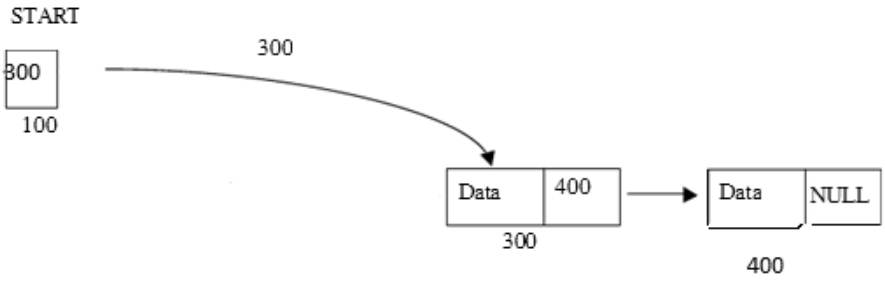

Fig.: After deleting node at beginning (address: 200)

ii) Deleting element at given node:

For deleting element at given node. Enter node value. Then the node and data elements are deleted. See below pictorial representation for better understanding.

Fig.: Initial Linked List

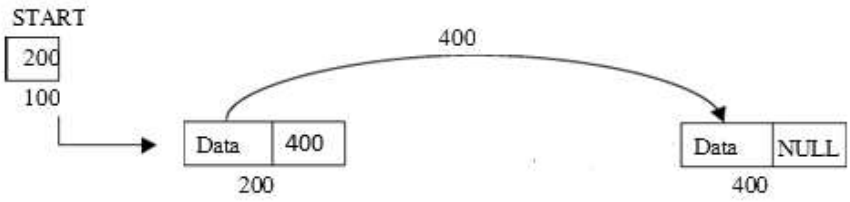

Fig.: After deleting node at given location (Examples LOC address 300)