Linked list

Date / /
Page

Standard blueprints

①
```
class Node
{
  public:
    int data;
    Node * next;
}
```

②
```
struct Node
{
    int data
    struct Node * next;
}
```

```
Node* new_node = new Node();
```

↳ way for ① & ② for creating an object of type Node and assigning memory dynamically.

*** ③**
```
class Node or Struct Node
{
  public:
    int data;
    Node *next;
    Node(int x)
    {
      data = x;
    }
}
```

```
Node * new_node = new Node(item);
```

↙ creates object dynamically and constructor is invoked and data for that object is inserted at that instant only.

→ constructor – No return type (invoked automatically when object is called)

Date / /
Page

Insertion at the beginning

```
/* Since return type of function we defined is void
   we have to pass head by reference that is
   in the form of (pointer to pointer). Along with it
   the data to be inserted has to be passed.
*/
Void insertAtBeginning(Node **head_ref, int new_data)
{
    /*1. Allocate node */
    Node * new_node = new Node();

    /* 2. Put in the data */
    new_node -> data = new_data;

    /* 3. Make next of new_node as node pointed by head */
    new_node -> next = *head_ref;

    /* 4. Make head point to new_node
    *head_ref = new_node;
}
```

Time complexity = O(1)

Node **head_ref ——reference to——→ Node *head;

Date / /
Page

RANKA

Add a node after given node

```
/* Given a node prev_node. Insert a new node
   after the given previous prev_node
*/
Void insertAfter( Node * prev_node, int new_data)
{
   /*1. check if previous node is NULL */
   if (prev_node == NULL)
   {
      cout<< "The given previous node cannot be NULL";
      return;
   }
   /* 2. Allocate new node */
   Node * new_node = new Node();
   new_node -> data = new_data
   /* 3. Make next of new_node point to next of prev_node */
   new_node ->next = prev_node ->next;
   /* 4. Make prev_node point to new_node */
   prev_node ->next = new_node;
}
```

→ O(1) as prev_node is passed to function already.

Date / /
Page

RANKA

Insert new node at the end

```
Void insertAtEnd ( Node** head_ref, int new_data )
{
    Node *new_node = new Node();
    new_node → data = new_data;
    new_node → next = NULL

    Node * last = *head_ref;
    if (*head_ref == NULL)
    {
      *head_ref = new_node;
       return;
    }
    while (last → next != NULL)
    {
        last = last → next;
    }

    last → next = new_node;
}
```

Time complexity = O(n)