Dept of Electronics &Communications **First Year**
Data Structure **sheet #5 lists**
Instructor: Prof. Neamat Abdelkader

---

1) Using the linked_list class, what will be the output of the following program:

```
int main ( )
{ linked_list L, B;
  elemtype x , y;  int n =5;
   for( int i=0; i <n ; i++)
    {   L. insert (i+1);      1 2 3 4 5
        B.insert(i*3); }      0 3 6 9 12
   cout <<" elements of the list  L"<<endl;
   bool found=L.first(x);
   while (found){ cout<<x<<"    "; found= L. next(x);}
    cout<<endl;
   found= B.first(y);
    while (found) {L.insert(y); found=B.next(y);}
     found=L.first(x); cout <<" elements of the list  L"<<endl;
     while (found){ cout<<x<<"    "; found= L. next(x);} return 0;}
```

elements of list L
1 2 3 4 5
elemnets of the list L
1 2 3 4 5 0 3 6 9 12

append

link added_ node (new node) or

link added_ node = new node

2) Write the external function split( ) that uses the linked_list class to split a list L into two lists L1 and L2, coressponding to certain target where, L1 containes the elements greater than the target and L2 contains the other elements.

اسال قبل ما تدخل while(curr != 0) or for(;curr != 0; . . .)

3) Write a function that counts the number of the nodes in a linked list.

4) Using the class linked list, what will be the output of the followng program if:
n=5 ; the values of x are(1, 5, 7, 10, 15)

```
int linked_list::get( )
{  current =head; int sum=head->elem;
   while( current->next != NULL)
    {sum+= current->next->elem;
   current= current ->next;}
   return sum;}

void solve(linked_list  &L)
{ elemtype x; bool found;
 found = L.first(x) ;
    while( found)
 {cout<<x<<"   "<<endl;   found =L.next(x);  found=L.next(x);  }}

int main ( )
{ linked_list L; elemtype x ;  int n;
     cin>>n;
     for( int i=0; i<n ; i++)      { cin>>x; L. insert (x); }
  int s=L. get( );  cout<<s<<endl;   solve(L);     return 0;}
```

38
1
7
15

```
void append (linked_list & l2)
{
   if(head == 0) head = l2.head;
   else          tail-> next = l2.head;
  tail = l2.tail;
}
```

5) To the standard linked list implementaion, add the following functions:

1. append(List A) to append list A at the end of this list(L)

2. List* L.copy() makes a copy of this list L and returns a pointer to it.

6) Add the reverse member function to the standard linked list that reverses the list in another list. If the list is empty, do nothing. To reverse a non-empty list, each node should point to the node that was privously its predecessor, the header should point to the last node, and the node that was formerly first should have a null link.

7) To the standard linked list, add a function Remove to delete any item from the linked list and returns the deleted node to the memory. Then use the class linked list to delete all odd elements(first, third, fifth,...).

8) Add the following member functions to the class double linked_list:
a. sort ( ) function to sort the elements of the list.
b. remove function to search for certain element and delete it from the list
Write a program that uses the double linked list class and reads n integer numbers and inserts them to the list, then use the sort function to sort the numbers and print the list after sorting.

9) Write a member function "insertBefore" that will insert an element before the current node of a linked list, make the necessary correction to the list.

10) Add the Function Append (list L1) to the double linked list to append the list L1 at the end of this list.

11) Use the **double linked list class** and add the member function min_elem( ) to get the minimum element and put it at the head.