CMPE-231 DATA STRUCTURES

••• Laboratory Work # 6

Duration – 100 minutes

This laboratory work covers implementation of Linked list structures.

Linked List

REMINDER:

- A linked list is a dynamic data structure,
- The pointer to a linked list – that is, the pointer to the first node in the list – is stored in a separate location,
- A (singly) linked list is traversed in only one direction,
- Linked lists are collections of data items "lined up in a row" – insertions and removals are made anywhere in a linked list,
- The dynamic nature of a list may be contrasted with the static nature of an array, whose size remains constant,
- Linked lists are important not only as a means of implementing stacks and queues but as data structures in their own right,
- An item is accessed in a linked list by traversing the list from its beginning [5,10,15],
- We can use a linked list to create *linear* and *non-linear structures*. In linear linked lists, each element has only zero or one successor. In non-linear lists, each element can have zero, one, or more successors. ... The major advantage of the linked list over the array is that data are easily inserted and deleted

1) The following programs can be used for self-studying for linked list **insertion sort** operation:

```
#include<stdlib.h>
#include <stdio.h>
#include <string.h>
#define NULL 0
struct node
 {
   char info[10];
   struct node *n;
  };
void main(void)
{
typedef struct node *NODEPTR;
char a[10];
int cnt=0;
 NODEPTR x , y , save , head , p , q;
        x = (NODEPTR)malloc(sizeof(struct node));
        y = (NODEPTR)malloc(sizeof(struct node));
        head=x;
```

```
        x->n=y;
        y->n=NULL;

        strcpy(x->info,"cemal");
        strcpy(y->info,"mert");

do
{
  puts("Enter Name information");
  gets(a);
  cnt++;
  q=NULL;
  for(p=head; p!=NULL && strcmp(a,p->info)>0; p=p->n)
    q = p;
   if (q == NULL) {
                        p=(NODEPTR)malloc(sizeof(struct node));
                        strcpy(p->info,a);
                        p->n = head;
                        head=p;
                        }
                else
                {
                  save=p;
                  p=(NODEPTR)malloc(sizeof(struct node));
                  strcpy(p->info,a);
                  p->n=save;
                  q->n=p;
                  }
  }while (cnt!=3);

 for(save=head;save!=NULL;save=save->n)
   printf("Traverse Node =%s\n",save->info);
getchar();
}
```

2) Do some modification in the program that will create the linked list in **descending** order and list the output.

3) Complete above program in a way that **will delete any given person(name will be taken from the keyboard) from the linked list structure. List content of linked list after deletion.**

Note: Give error message if the given person is not in the linked list.