KING SAUD UNIVERSITY COLLEGE OF COMPUTER AND INFORMATION SCIENCES COMPUTER SCIENCE DEPARTMENT		
CSC215	Lab10 Linked Lists	

Write a program that creates in the main function two linked lists of characters and fills them with the following values:

- The first list will have 3 nodes with the following characters: A,B, and C.
- The second list will also have 3 nodes with the following characters: D,E, and F.

In your main also:

- Call the function concat and send your first and second lists as arguments.
- Call the function printList and send your concatenated list as argument.
- Call the function addNode and send your concatenated list and the value 'G' as arguments.
- Call the function printList again and send your concatenated list as argument to print the content of the list after adding a node.

Write a function that concatenates two linked lists of string. The function takes pointers to both lists as arguments and concatenates the second list to the first list.

- void concat (struct node *f, struct node *s)
- In your function consider the cases where the first or second lists are empty.

Write a function that adds a new node to the end of the list. The function takes a pointer to the concatenated list and the new node value. Your function should print feedback messages upon successful/ unsuccessful addition.

- `void addNode(struct node *head, char val)`

Write a function that prints the content of the concatenated list. The function takes a pointer to the concatenated list.

- `void printList(struct node *head)`

Your output after running your program should be:

```
The Content of the Concatenated List is:
A
B
C
D
E
F
The node was added successfully
After adding a node:
The Content of the Concatenated List is:
A
B
C
D
E
F
G
```

Model Answer:

```
# include <stdio.h>

# include <stdlib.h>



struct node {

char val;

struct node *next;
};


void concat ( struct node *f, struct node *s )

{

struct node *temp ;


/* if the first linked list is empty */

if ( f == NULL )

*f = *s ;

else

{

/* if both linked lists are non-empty */

if ( s != NULL )

{
```

```
 temp = f ; /* points to the starting of the first list*/

/* traverse the entire first linked list */

while ( temp -> next != NULL )

temp = temp -> next ;

/* concatenate the second list  after the first */


 temp -> next = s ;

  }

  }

  }
//end concat

void addNode(struct node *head, char val){



struct  node * current = head;

while (current->next != NULL) {

        current = current->next;

    }

    /* now we can add a new node */

    current->next = malloc(sizeof(struct node));
    if(current != NULL)
    {

    current->next->val = val;

    current->next->next = NULL;

    printf("The node was added successfully");
    }
    else
    printf("No enough memory to add a node");

}
```

```
//end addNode

void printList(struct node *head)
{
struct node *current = head;

  printf("The Content of the Concatenated List is: \n");

  while(current != NULL)

  {

  printf("%c\n",current->val);

  current = current ->next;
}


  }//end printList



  int main( )

  {

  struct node *first, *second , *head1, *head2;

  int i;

  first = second = head1 = head2 = NULL ;  /* empty
linked lists */

  char C = 'C';
  for (i = 0; i<3; i++)

  {

  first = (struct node*)malloc(sizeof(struct node));

  if(first != NULL)

 {
  first -> val =C;

  first -> next = head1;

  head1 = first;
```

```
  C--;
  }

 }

  char F = 'F';


  for (i = 0; i<3; i++)

  {

  second = (struct node*)malloc(sizeof(struct node));

  if(second != NULL)

  {

  second -> val =F;

 second -> next = head2;

  head2 = second;
  F--;

  }

  }
//calling functions
  concat(first, second);
  printList(head1);
  addNode(head1, 'G');
  printf("\nAfter adding a node:\n");
  printList(head1);
  return 0;

  }//end main
```