7-12-20
Ankit Prasad
1BM19CS192

Lab-7/8:- Reverse, Sort & Concat, Stack & Queue.

→ Concatenation

```
void concatenate (struct node *a, struct node *b)
{
   if (a != NULL && b != NULL)
{
   if (a->next, b);
 }
 else
 {
     printf ("Either a or b is NULL \n");
 }
}
struct node * concat (struct node * start1, struct node *start2)
   {
     struct node *ptr;
     if (start1 == NULL)
 {
     start1 = start2;
     return start1;
 }
   if (start2 == NULL)
        return start1;
ptr = start1;
while (ptr -> link != NULL)
        ptr = ptr -> link;
ptr -> link = start2;
  return start1;
}
```

Reverse:

```
while (current != NULL)
    {
        next = current -> next ;
        current -> next = prev;
        prev = current ;
        current = next ;
    }
    *head_ref = prev;
```

Sort:

```
void sortlist ( ) {
    struct node * current = head , * index = NULL;
    int temp ;
if (head == NULL) {
        return ;
    }
else {
    while (current != NULL ) {
        index = current -> next ;
    while (index != NULL ){
        if (current -> data > index -> data ) {
    temp = current -> data ;
        current -> data = index -> data ;
        index -> data = temp ;
    }

    } index = index -> next;

} } } current = current -> next
```

Stacks :-

```
void push (int i, element item)
  {
     stack pointer temp;
   malloc (temp, sizeof (*temp));
 temp -> data = item;
    temp -> link = top [i]
 top [i] = temp;
  }
 void pop (int i)
{
 stack pointer temp = top [i]
 element item;
  if (!temp)
 return stack Empty ();
item = tem -> data
 top [i] = temp -> be;
free (temp);
return item;
}
```

Queue :

```
front [i] = NULL, 0 ≤ i ≤ MAX;
    front [i] = NULL;
void addq (i, item)
    {
       queue pointer temp;
malloc (temp, sizeof (*temp));
```

```
temp -> data = item ;
temp -> link = NULL ;
    if (front[i])
rear[i] -> temp;
  else
   front[i] = temp;
rear[i] = temp.

void deleteq (int i) {  1" Delete from the front
of a linked queue

    temp = front[i];
element item ;
  if (!temp)
return queue Empty();
item = temp -> data;
   front[i] = temp -> link;
free(temp);
  return item ;
}
```