TASMIYA FATHIMA
1BM19CS172

LAB-8

WAP Implement single link list with following operations

a) sort the linked list
b) Reverse the linked list
c) Concatenation of two linked lists
d) implement stack & queues using linked Representation

```
NODE * sort (NODE *S)
{
  NODE *t1, *t2
  int temp;
for( t1= S; t1-> Link! = NULL; t1 = t1-> link)
{
for (t2 = t1 -> link; t2! = NULL ; t2 = t2-> link )
{
  if ( t1 -> info > t2 -> info )
{
    temp = t1 -> info ;
    t1 -> info = t2 -> info ;
    t2 -> info = temp ;
};
}
}
```

Reversing

```
NODE * rev ( NODE * start)
{
  NODE *t1, t2, * S;
  for( t1 = start ; t1 -> link ! = NULL ; t1 = t1->link)
    s = t1;
    while (t1! = start)
{
  for (t2 = start ; t2 -> link ! = t1; t2 = t2->link)
```

```
t1 -> link = t2;
t1 = t2;
}
return S;
}
```

Concatenation

```
NODE * concat (NODE *S1, NODE *S2)
{
  NODE *t;
  t = S1;
  while (t -> link != NULL)
  t = t -> link;
  t -> link = S2;
  return S1;
}
```

Display

```
void disp (NODE *start)
{
  NODE *t;
  for (t = start; t != NULL; t = t -> link)
  {
    if (t -> link != NULL)
    printf ("%d", t -> info)
    else
    printf ("%d", t -> info)
  }
}
```

pushing into stack

```
Void push()
{
int info;
NODE *ptr = (NODE*) malloc (sizeof(NODE));
if( ptr == NULL)
printf(" Empty\n");
else
{
scanf("%d", &info);
if (head == NULL)
{
ptr -> info = info
ptr -> link = NULL;
head = ptr;
}
else
{
ptr -> info = info
ptr -> link = head;
head = ptr;
}}}
```

Popping from stack

```
Void pop()
{
int item;
NODE *ptr;
if( head == NULL)
printf(" Underflow \n");
else
{
```

```
item = head -> info ;
ptr = head ;
head = head -> link ;
free (ptr) ;
}
}
```

Enqueue

```
void enqueue ()
{
  NODE * ptr ;
  int item ;
  ptr = (NODE *) malloc (size of (NODE)) ;
  if ( ptr == NULL )
  {
    printf (" Overflow \n") ;
    return ;
  }
  else
  {
    scanf ("%d", & item) ;
    ptr -> info = item ;
    if ( front == NULL)
  {
    front = ptr ;
    rear = ptr ;
    rear -> link = NULL ;
    front -> link = NULL ;
  }
else
{
  rear -> link = ptr ;
  rear = ptr ;
```

```
rear -> link = NULL;
}
}
}
```

dequeue

```
Void dequeue ()
{
 NODE * ptr ;
 if ( front == NULL)
{
  printf (" Underflow \n");
  return ;
}
 else
 {
   ptr = front ;
   front = front -> link ;
   free (ptr);
 }
}
}
```