```
void reverse ( )
{ struct node *prev, * current, *next;
  current = head
  prev = NULL
  next = NULL
  while (current != NULL)
  { next = current->next
    current->next = prev
    prev = current
    current = next
  }
  head = prev
}
```

```
struct node * concatenate (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->next != NULL)
        ptr = ptr->link
ptr->link = start2
 return start1
}
```

```
void sort()
{ int flag, i;
  struct node *ptr1, *ptr = NULL
  if(head==NULL)
  { print "List is Empty"
  return }
  do
    { flag=0;
      ptr1=head
      while(ptr1->next!=ptr)
      { if(ptr1->data > ptr1->next->data)
        { swap(ptr1, ptr1->next)
          flag=1
        }
        ptr1=ptr1->next } ptr=ptr1
      } while(flag==1)
  print "Swapped"
}

void swap(node *a, node *b)
{ int temp = a->data
  a->data = b->data
  b->data = temp
}
```

Date __/__/__
Page
SPLASH

Stack implementation

```
Void push (struct node*
 void push ( )
{  struct node *new_node = (struct node*) malloc (sizeof (struct node))
   new_node -> data = new_item
   new_node -> next = head
   head = new_node
}

 void Pop ( )
{ struct node *ptr
   if (head == NULL)
  print "List is Empty"
  else
    { ptr = head
      head = ptr -> next
       free (ptr)
     print "Node deleted"
    }
}
```

Queue implementation

```
void Enqueue (int item)
{ struct node *ptr, *temp
  ptr = (struct node *) malloc ( sizeof (struct node));
  ptr->data = item
  ptr->next = NULL
  if (head = NULL)
  { head = ptr }
else
  { temp = head
    while (temp->next != NULL)
   temp = temp->next
   temp->next = ptr
    }
}

  void Dequeue ()
{ struct node *ptr
    if (head == NULL)
    print "Linked List is Empty"
else
    { ptr = head
      ptr head = ptr->next
      free (ptr)
      print "Element deleted"
    }
}
```