Q) WAP to implement single linked list with following operations :-

(i) Sort the linked list.

(ii) Reverse the linked list.

(iii) Concatenation of 2 linked list.

(iv) Implement Stack & queue using linked representation.

Ans:

```
#include <stdio.h>
#include <stdlib.h>

struct Node { int data;
              struct Node * next;
            };

void bubbleSort (struct Node *start)
{ int swapped, i;
  struct Node *ptr1;
  struct Node *lptr = NULL;
  if (start == NULL)
   return;
  do
   { swapped = 0;
     ptr1 = start;
     while (ptr1 -> next != lptr)
      { if (ptr1 -> data > ptr1 -> next -> data)
         { swap (ptr1, ptr1 -> next);
           swapped = 1;
         }
        ptr1 = ptr1 -> next;
      }
     lptr = ptr1;
   } while (swapped);
}
```

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

void concatenate (struct Node *a, struct Node *b)
{ if (a != NULL && b != NULL)
    { if (a -> next == NULL)
        a -> next = b;
      else
        concatenate (a -> next, b);
    }
    else
      { printf(" either a or b is NULL "); }
}

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;
}
```

```
// stack implementation.
void push (struct Node **head_ref, int new_data)
{ struct Node *new_node = (struct Node*) malloc (sizeof (struct Node));
  new_Node->data = new_data;
  new_Node->next = (*head_ref);
  (*head_ref) = new_Node;
}

void pop()
{ struct Node *ptr;
  if (head == NULL)
  { printf ("List is empty");
  }
  else
  { ptr = head;
    head = ptr->next;
    free (ptr);
    printf ("Node deleted from beginning");
  }
}

// Queue implementation.
void Enqueue (item)
{ struct Node *ptr, *temp;
  ptr = (struct Node*) malloc (sizeof (struct Node));
  ptr->data = item;
  ptr->next = NULL;
  if (head == NULL)
  { head = ptr;
    printf("Node inserted");
  }
  else
  { temp = head;
```

```
while (temp→next != NULL)
  { temp = temp→next; }
  temp → next = ptr;
  printf(" Node Inserted");
 }
}

void Dequeue ()
{ struct Node *ptr;
  if (head == NULL)
  { printf (" List is empty");
  }
  else
  { ptr = head;
    head = ptr → next;
    free (ptr);
    printf(" Node deleted from beginning");
  }
}
```

———— o ————