3. Single linked list - sort, reverse, concatination

```
#include<stdio.h>
#include<stdlib.h>
struct node
{ int data;
  struct node *next;
};
void insert(struct node **head, int data)
{ struct node* newnode = (struct node*) malloc
                         (sizeof(struct node));
  newnode -> data = data;
  newnode -> next = *head
  *head = newnode
}
void print(struct node *head)
{ while(head != NULL)
  { printf("%d \t", read->data);
    read = head -> next;
  }
}
void sort (struct node **head)
{ struct node *current, *newnode;
  int temp;
  current = *head;
  while (current != NULL)
  { nextnode = current -> next;
    while (nextnode != NULL)
    { if (current -> data > nextnode -> data)
      { temp = current -> data;
        current -> data = nextnode -> data;
        nextnode -> data = temp;
      }
      nextnode = nextnode -> next;
    } current = current -> next; } }
```

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

Void concatenate(struct node **list1, struct node
                 **list2)
{
    if (*list1 == NULL)
      { *list1 = *list2;
        return;
      }
    Struct node *temp **list1;
     while (temp->next != NULL)
     { temp = temp->next;
     }
     temp->next = list2;
}
void main()
{
  struct node *list1 = NULL;
  struct node *list2 = NULL;
   int choice;
   int data;
   while(1)
   { printf("1. Insert into list1 \t, 2. Insert
into list2 \t 3. Sort list1 \t 4. Concatenate
lists \t 5. Reverse list1 \t 6. Display \t
        7. Exit \n");
```

```
printf("Enter your choice ");
scanf("%d", &choice);
switch(choice)
{
case 1 : printf("Enter data to insert into list1:");
         scanf("%d", &data);
         insert(&list1, data);
         break;
Case 2 : printf("Enter data to insert in list2:");
         scanf("%d", &data);
         insert(&list2, data);
         break;
Case 3 :
         sort(&list1);
         printf("List 1 Sorted");
         break;
Case 4 : concatenate(&list1, &list2);
         printf("lists concatinated");
         break;
Case 5 :
         reverse(&list1);
         printf("list1 reversed");
         break;
Case 6 : printf("list1:");
         print(list1);
         break;
Case 7 :
         exit(0);
         break;
default: printf("Invalid Input");
}
}
}
```