Programme - 7

WAP Implement Single Link List with following operations.

a) Sort the linked list.
b) Reverse the linked list
c) concatenation of two linked list.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE) malloc(sizeof(struct node));
if(x==NULL)
{
printf("memory full\n");
exit(0);
}
return x;
}
```

```
void freenode (NODE x)
{
 free (x);
}
NODE insert-front(NODE first, int item)
{
NODE temp;
temp = getnode ();
temp -> info = item;
temp -> link = NULL;
if (first = = NULL)
 return temp;
temp -> link = first;
first = temp;
return first; }

NODE delete-front (NODE first){
 NODE temp;
 if (first = = NULL)
{ printf ("list is empty cannot delete \n");
  return first; }
temp = first;
temp = temp -> link;
printf ("item deleted at front - end is = %d \n",
                    first -> info);

NODE reverse (NODE first)
{
 NODE cur, temp;
cur = NULL;
while (first != NULL)
{ temp = first;
```

```
first = first -> link;
temp -> link = cur;
cur = temp;
}
return cur;
}
NODE concat (NODE first, NODE second)
{ NODE cur;
  if (first == NULL)
    return second;
  if (second == NULL)
    return first;
  cur = first;
 While(cur-> link != NULL)
  Cur = cur -> link;
 Cur -> link = second;
 ~~second~~ return first; }
void display(NODE first)
{ NODE temp;
 if ( first == NULL)
 printf ("list empty cannot display items\n");
 for(temp = first; temp != NULL; temp = temp -> link)
 printf ("%d\n", temp -> Info);
}
NODE ~~E~~ sortList(NODE first){
 NODE current = first, index = NULL;
 int temp;
 if ( first == NULL)
{ printf ("list is empty");
    return current; }
else{ while (current != NULL){
         index = current -> link;
```

```
while (index != NULL){
if (current -> info > index->info) {
    temp = current -> info;
     current -> info = index -> info;
     index -> info = temp;
  } index = index -> link;
 } current = current -> link;
 } return current;
 } }
Void main(){
 int item, choice, pos, n, i;
 NODE first = NULL, a, b;
for(;;)
{ printf ("\n 1: Insert front 2: Delete front 3: reverse list
          4: Concat list 5: sort list 6: display list
          7: Exit \n");
printf ("%d", &choice);
Switch (choice)
{
 case 1: printf ("enter the item at front end\n");
 scanf ("%d", &n);
    ~~a = NULL;~~
 first = insert_front (first, item);
 break;
Case 2: first = delete_front(first);
break;
Case 3: first = ~~delete_front~~(first); reverse(first); display(first);
break;
case 4:
  if (first == NULL) {
    printf ("enter the no of nodes in 1:");
     scanf ("%d", &n);
a = NULL;
for( i=0 ; i<n; i++)
  {
     printf ("enter the item:");
     scanf ("%d ", &item);
     a = insert_front(a, item);
  }} else {
     a = first; }
     printf ("enter the no of nodes in list:");
     scanf("%d", &n);
     b = NULL;
     for(i=0; i<n; i++)
     { printf ("enter the item:");
       scanf ("%d", &item);
       b = insert_front (b, item);
     }
       a = concate (a,b);
          display (a);
        break;
case 5 : sort_list (first);
         display(first);
        break;
case 6 : display(first);
        break;
default : ~~for(;~~ first - exist(0);
}}
}
```

st

first);