LAB PROGRAM-7 AND 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) Stack and Queue Implementation.

CODE:

```
#include<stdio.h>
#include<stdlib.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("mem 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 insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE insert_pos(int item,int pos,NODE first)
{
NODE temp;
NODE prev,cur;
int count;
temp=getnode();
temp->info=item;
```

```
temp->link=NULL;
if(first==NULL && pos==1)
return temp;
if(first==NULL)
{
printf("invalid pos\n");
return first;
}
if(pos==1)
{
temp->link=first;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL && count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("IP\n");
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);
free(first);
return temp;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
```

```
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE delete_pos(int pos,NODE first)
{
NODE prev,cur;
int count;
if (first==NULL || pos<=0)
{
printf("Invalid position\n");
return NULL;
}
if (pos==1)
{
cur=first;
first=first->link;
printf("Item deleted at position %d is %d",pos,cur->info);
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while (cur!=NULL)
{
if (count==pos)
{
break;
}
prev=cur;
cur=cur->link;count++;
```

```
}
if (count!=pos)
{
printf("Invalid position\n");
return first;
}
prev->link=cur->link;
printf("Item deleted at position %d is %d",pos,cur->info);
freenode(cur);
return first;
}
NODE order_list(int item,NODE first)
{
NODE temp,prev,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL) return temp;
if(item<first->info)
{
temp->link=first;
return temp;
}
prev=NULL;
cur=first;
while(cur!=NULL&&item>cur->info)
{
prev=cur;
cur=cur->link;
}
prev->link=temp;
temp->link=cur;
return first;
```

```
}
NODE sort(NODE first)
{
int swapped;
NODE ptr1;
NODE lptr = NULL;
if (first == NULL)
return NULL;
do
{
swapped = 0;
ptr1 = first;
while (ptr1->link != lptr)
{
if (ptr1->info > ptr1->link->info)
{
int tem = ptr1->info;
ptr1->info = ptr1->link->info;
ptr1->link->info = tem;
swapped = 1;
}
ptr1 = ptr1->link;
}
lptr = ptr1;
} while (swapped);
}
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;
return first;
}
NODE reverse(NODE first)
{
NODE cur,temp;
cur=NULL;
while(first!=NULL)
{
temp=first;
first=first->link;
temp->link=cur;
cur=temp;
}
return cur;
}
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);
}
}
int length(NODE first)
{
NODE cur;
int count=0;
```

```
if(first==NULL) return 0;
cur=first;
while(cur!=NULL)
{
count++;
cur=cur->link;
}
return count;
}
void search(int key,NODE first)
{
NODE cur;
int count1=0;
if(first==NULL)
{
printf("List is empty\n");
return;
}
cur=first;
while(cur!=NULL)
{
count1++;
if(key==cur->info)
break;
cur=cur->link;
}
if(cur==NULL)
{
printf("Search is unsuccessful\n");
return;
}
printf("Search is successfull\n");
printf("Item present at the position number %d\n",count1);
```

```
}
void main()
{
int item,choice,pos,i,n,count,key;
NODE first=NULL,a,b;
for(;;)
{
printf("\n 1:Insert_front\n 2:Insert_rear\n 3:Insert_pos\n
4:Delete_front\n 5:Delete_rear\n 6:Delete_pos\n 7:Sort_list\n
8:Order_list\n 9:Concat\n 10:Reverse List\n 11:Display_list\n
12:Stack\n 13:Queue\n 14:Length of the list\n 15:Search item\n
16:Exit\n");
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 3:printf("enter the position\n");
scanf("%d",&pos);
printf("Enter the item\n");
scanf("%d",&item);
first=insert_pos(item,pos,first);
break;
case 4:first=delete_front(first);
break;
case 5:first=delete_rear(first);
```

```
break;
case 6:printf("Enter the position:\n");
scanf("%d",&pos);
first=delete_pos(pos,first);
break;
case 7:sort(first);
break;
case 8:printf("Enter the item to be inserted in ordered_list\n");
scanf("%d",&item);
first=order_list(item,first);
break;
case 9:printf("Enter the no of nodes in 1\n");
scanf("%d",&n);
a=NULL;
for(i=0;i<n;i++)
{
printf("Enter the item\n");
scanf("%d",&item);
a=insert_rear(a,item);
}
printf("Enter the no of nodes in 2\n");
scanf("%d",&n);
b=NULL;
for(i=0;i<n;i++)
{
printf("Enter the item\n");
scanf("%d",&item);
b=insert_rear(b,item);
}
a=concat(a,b);
printf("\n");
printf("Items are :\n");
display(a);
```

```
break;
case 10:first=reverse(first);
printf("Items of the reverse list are :\n");
display(first);
break;
case 11:display(first);
break;
case 12:printf("Stack\n");
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_rear\n 3:Display_list\n
4:Exit\n");
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_rear(first);
break;
case 3:display(first);
break;
default:exit(0);
break;
}
}
case 13:printf("QUEUE\n");
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n
4:Exit\n");
```

```
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
default:exit(0);
break;
}
}
case 14:count=length(first);
printf("Length(items) in the list is %d\n",count);
break;
case 15:printf("Enter the item to be searched\n");
scanf("%d",&key);
search(key,first);
break;
default:exit(0);
break;
}
}
getch();
}
```

OUTPUT :

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
20
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
2
enter the item at rear-end
30

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
2
enter the item at rear-end
40
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
20
10
30
40

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
3
enter the position
3
Enter the item
50
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
20
10
50
30
40

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
4
item deleted at front-end is=20
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
5
iten deleted at rear-end is 40
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
10
50
30
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
6
Enter the position:
2
Item deleted at position 2 is 50
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
10
30
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
40
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
40
20
10
30
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
7

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
10
20
30
40
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
8
Enter the item to be inserted in ordered_list
25

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
10
20
25
30
40
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
10
Items of the reverse list are :
40
30
25
20
10
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
14
Length(items) in the list is 5
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
15
Enter the item to be searched
25
Search is successfull
Item present at the position number 3
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
15
Enter the item to be searched
60
Search is unsuccessful
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
9
Enter the no of nodes in 1
2
Enter the item
10
Enter the item
20
Enter the no of nodes in 2
3
Enter the item
30
Enter the item
40
Enter the item
50

Items are :
10
20
30
40
50
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
12
Stack

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
50

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
40
30
25
20
10
50
```

```
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
60

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
iten deleted at rear-end is 60
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
40
30
25
20
10
50

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
4
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
11
40
30
25
20
10
50
```

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
 5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
 10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
 14:Length of the list
 15:Search item
 16:Exit
Enter the choice
13
QUEUE

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
60

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
40
30
25
20
10
50
60
```

```
 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
item deleted at front-end is=40

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
30
25
20
10
50
60

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
4

Process returned 0 (0x0)   execution time : 102.247 s
Press any key to continue.
```