Week -10

1. WAP to Implement Singly Linked List with following operations

a) Create a linked list.b) Insertion of a node at first position, at any position and at end of list.c) Display the contents of the linked list.

a) Create a linked list.

b) Deletion of first element, specified element and last element in the list.

c) Display the contents of the linked list.

```
#include<stdio.h>
#include<malloc.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("Memory full\n");
 exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}


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 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("Item deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}


NODE insert_pos(int item,int pos,NODE first)
{
NODE temp,cur,prev;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL&&pos==1)
{
return temp;
}
if(first==NULL)
{
printf("Invalid position\n");
return first;
}
if(pos==1)
{
temp->link=first;
first=temp;
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("Invalid position\n");
return first;
}


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

```
return first;
}


void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List is empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
 {
 printf("%d\n",temp->info);
 }
}


int main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_rear\n");
printf(" 3:Insert_info_position\n 4:Delete_info_position\n 5:Display_list\n 6:Exit\n");
printf("Enter the choice : ");
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:printf("Enter the item to be inserted at given position:\n");
 scanf("%d",&item);
 printf("Enter the position:\n");
 scanf("%d",&pos);
 first=insert_pos(item,pos,first);
 break;
 case 4:printf("Enter the position:\n");
 scanf("%d",&pos);
 first=delete_pos(pos,first);
 break;
```

```
 case 5:display(first);
 break;
 default:exit(0);
 break;
 }
}
return 0;
}
```

```
 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
10

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
20

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
40

 1:Insert_rear
```

```
 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 3
Enter the item to be inserted at given position:
30
Enter the position:
3

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 5
10
20
30
40

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice :
```

a) Create a linked list.
b) Deletion of first element, specified element and last element in the list.
c) Display the contents of the linked list.

```
 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
10

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
20

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
40

 1:Insert_rear
```

```
Enter the choice : 1
Enter the item at rear-end:
20

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
40

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
3
Item deleted at given position is 40

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice :
```

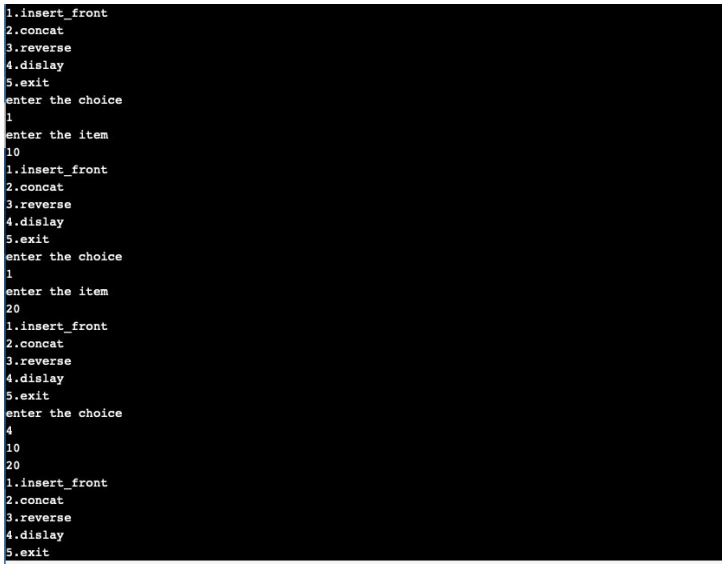
3.WAP Implement Single Link List with following operations

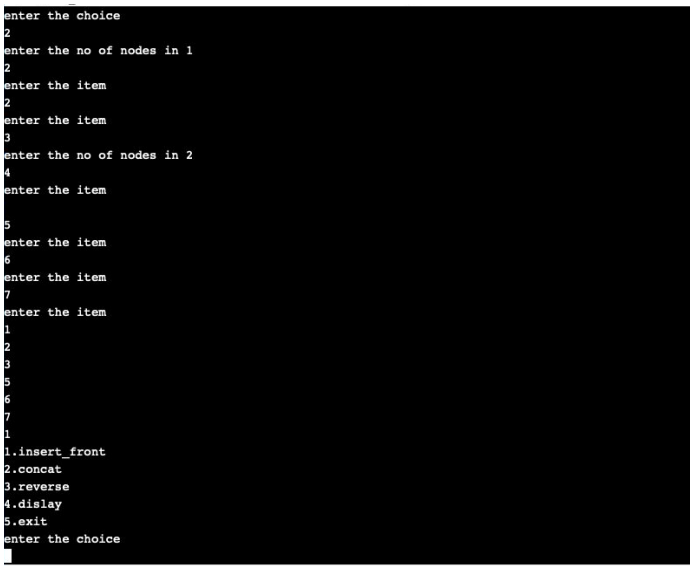
a) Sort the linked list.
b) Reverse the linked list.
c) Concatenation of two linked lists

```
#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;
}
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;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
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;
}
int main()
{
int item,choice,pos,i,n;
NODE first=NULL,a,b;

for(;;)
{
printf("1.insert_front\n2.concat\n3.reverse\n4.dislay\n5.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
```

```
case 2: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);
display(a);
break;
case 3:first=reverse(first);
display(first);
break;
case 4:display(first);
break;
default:exit(0);
}
}

}
```





4. WAP to implement Stack & Queues using Linked Representation

```
#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_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 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;
}
```

```
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 main()
{
int item,choice,pos;
NODE first=NULL;

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;
}
}

}
```

```
 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
list is empty cannot delete

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
list empty cannot display items

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at rear-end
6

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
```

```
enter the item at rear-end
9

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

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
6
9
3

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=6

 1:Insert_rear
 2:Delete_front
```

Stacks

```
#include<stdio.h>
// #include<conio.h>
// #include<alloc.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 delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("stack 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;
```

```
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("stack empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
int main()
{
int item,choice,pos;
NODE first=NULL;

for(;;)
{
printf("\n 1:Insert_front\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 front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
default:exit(0);
break;
}
}
}
```

```
 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
stack is empty cannot delete

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
```

```
enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
20
10

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=20

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
4


...Program finished with exit code 0
Press ENTER to exit console.
```