Week 8:(Linked lists)

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("Memory is 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("The item deleted from front of the list is : %d\n",first->info);
        free(first);
        return temp;
}
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\n",cur->info);
        free(cur);
        prev->link=NULL;
        return first;
}
void display(NODE first)
{
        NODE temp;
        if(first==NULL)
```

```
            printf("List is EMPTY!\n");
        for(temp=first;temp!=NULL;temp=temp->link)
        {
            printf("%d\n",temp->info);
        }
}
void main()
{
        int item,choice,pos;
        NODE first=NULL;

        for(;;)
        {

printf("\n-------------------------\n1:Insert_front\n2:Delete_front\n3:Insert_rear\n4:Delete_rear\n5:Display
_list\n6: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:printf("Enter the item at rear-end\n");
                        scanf("%d",&item);
                        first=insert_rear(first,item);
                        break;
                case 4:first=delete_rear(first);
                        break;
                case 5:
                        printf("The list is:\n");
                        display(first);
                        break;
                case 6: exit(0);break;
                default:printf("INVALID CHOICE!\n");
                        break;
            }
        }
}
```

Output:

```
C:\Users\misaf\Desktop\DS LA

-------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
5
The list is:
List is EMPTY!

-------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
1
Enter the item at front-end
10
```

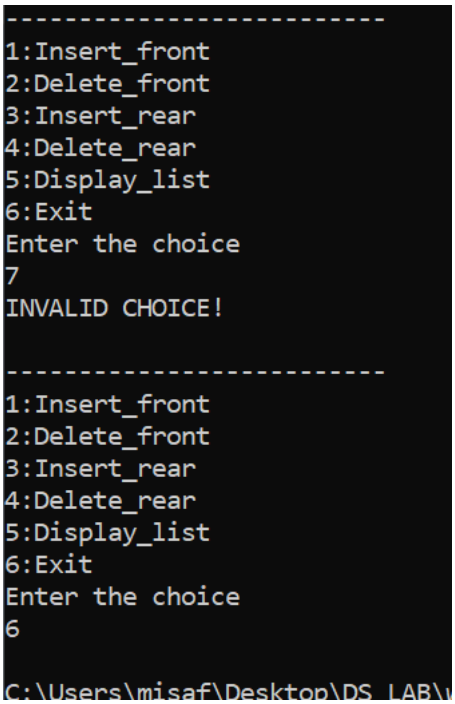
```
-------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
1
Enter the item at front-end
5

-------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
3
Enter the item at rear-end
15

-------------------------
```

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

-------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
5
The list is:
5
10
15
10
```

```
-------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
2
The item deleted from front of the list is : 5

-------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
2
The item deleted from front of the list is : 10

-------------------------
```

```
--------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
4
Item deleted at rear-end is 10

--------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
4
Item deleted is 15

--------------------------
```

```
--------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
2
List is empty cannot delete!

--------------------------
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
4
List is empty cannot delete

--------------------------
```


1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
7
INVALID CHOICE!
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Display_list
6:Exit
Enter the choice
6