```c
//Linked List Program
#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 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;
}
```

```
1843    }
1844    NODE insert_rear(NODE first,int item)
1845    {
1846        NODE temp,cur;
1847        temp=getnode();
1848        temp->info=item;
1849        temp->link=NULL;
1850        if(first==NULL)
1851            return temp;
1852        cur=first;
1853        while(cur->link!=NULL)
1854            cur=cur->link;
1855        cur->link=temp;
1856        return first;
1857    }
1858    NODE insert_pos(int item,int pos,NODE first)
1859    {
1860        NODE temp,prev,cur;
1861        temp=getnode();
1862        temp->info=item;
1863        temp->link=NULL;
1864        if(pos==1&&first==NULL)
1865        {
1866            return temp;
1867        }
1868        if(first==NULL)
1869        {
1870            printf("Invalid Positon\n");
1871            return first;
1872        }
1873        if(pos==1)
1874        {
1875            temp->link=first;
1876            return temp;
1877        }
1878        int count=1;
1879        cur=first;
```

```
1879        cur=first;
1880        prev=NULL;
1881        while(cur!=NULL&&count!=pos)
1882        {
1883            prev=cur;
1884            cur=cur->link;
1885            count++;
1886        }
1887        if(count==pos)
1888        {
1889            prev->link=temp;
1890            temp->link=cur;
1891            return first;
1892        }
1893        printf("Invalid Position\n");
1894        return first;
1895
1896    }
1897    NODE delete_front(NODE first)
1898    {
1899        NODE temp;
1900        if(first==NULL)
1901        {
1902            printf("list is empty cannot delete\n");
1903            return first;
1904        }
1905        temp=first;
1906        temp=temp->link;
1907        printf("item deleted at front-end is=%d\n",first->info);
1908        free(first);
1909        return temp;
1910    }
1911    NODE delete_rear(NODE first)
1912    {
1913        NODE cur,prev;
1914        if(first==NULL)
1915        {
```

```
1915        {
1916            printf("list is empty cannot delete\n");
1917            return first;
1918        }
1919        if(first->link==NULL)
1920        {
1921            printf("item deleted is %d\n",first->info);
1922            free(first);
1923            return NULL;
1924        }
1925        prev=NULL;
1926        cur=first;
1927        while(cur->link!=NULL)
1928        {
1929            prev=cur;
1930            cur=cur->link;
1931        }
1932        printf("item deleted at rear-end is %d",cur->info);
1933        free(cur);
1934        prev->link=NULL;
1935        return first;
1936    }
1937    NODE delete_pos(int pos, NODE first)
1938    {
1939        NODE cur;
1940        NODE prev;
1941        int count;
1942        if(first==NULL || pos<=0)
1943        {
1944            printf("invalid position\n");
1945            return NULL;
1946        }
1947        if (pos==1)
1948        {
1949            cur=first;
1950            first=first->link;
1951            freenode(cur);
```

```
1951            freenode(cur);
1952            printf("Node deleted successfully\n");
1953            return first;
1954        }
1955        prev=NULL;
1956        cur=first;
1957        count=1;
1958        while(cur!=NULL){
1959            if(count==pos){ break;}
1960            prev=cur;
1961            cur=cur->link;
1962            count++;
1963        }
1964        if(count!=pos)
1965        {
1966            printf("invalid position\n");
1967            return first;
1968        }
1969        if(count!=pos)
1970        {
1971            printf("invalid position specified\n");
1972            return first;
1973        }
1974        prev->link=cur->link;
1975        freenode(cur);
1976        return first;
1977    }
1978    void swap(NODE a,NODE b)
1979    {
1980        int temp = a->info;
1981        a->info = b->info;
1982        b->info = temp;
1983    }
1984    void bubbleSort(NODE first)
1985    {
1986        int swapped;
1987        NODE cur;
```

```
1987        NODE cur;
1988        NODE prev = NULL;
1989
1990        if (first == NULL)
1991        {
1992            printf("Empty Linked List\n");
1993            return;
1994        }
1995        do
1996        {
1997            swapped = 0;
1998            cur = first;
1999
2000            while (cur->link != prev)
2001            {
2002                if (cur->info > cur->link->info)
2003                {
2004                    swap(cur, cur->link);
2005                    swapped = 1;
2006                }
2007                cur = cur->link;
2008            }
2009            prev=cur;
2010        }
2011        while (swapped);
2012    }
2013    NODE concat(NODE first,NODE second)
2014    {
2015        NODE cur;
2016        if(first==NULL)
2017            return second;
2018        if(second==NULL)
2019            return first;
2020        cur=first;
2021        while(cur->link!=NULL)
2022            cur=cur->link;
2023        cur->link=second;
```

```
2023        cur->link=second;
2024        return first;
2025    }
2026    NODE reverse(NODE first)
2027    {
2028        NODE cur,temp;
2029        cur=NULL;
2030        while(first!=NULL)
2031        {
2032            temp=first;
2033            first=first->link;
2034            temp->link=cur;
2035            cur=temp;
2036        }
2037        return cur;
2038    }
2039    void display(NODE first)
2040    {
2041        NODE temp;
2042        if(first==NULL)
2043            printf("list empty cannot display items\n");
2044        for(temp=first;temp!=NULL;temp=temp->link)
2045        {
2046            printf("%d\n",temp->info);
2047        }
2048    }
2049    int main()
2050    {
2051        int item,choice,pos,i,n;
2052        NODE first=NULL,a,b;
2053        for(;;){
2054            printf("\n 1:Insert_front\n 2:Insert_rear\n 3.Insert at specified position\n 4:Delete_front\n 5.Delete_rear\n 6.Delete at specified position\n
2055            printf("enter the choice\n");
2056            scanf("%d",&choice);
2057            switch(choice)
2058            {
2059                case 1:printf("enter the item at front-end\n");
```

```
2059            case 1:printf("enter the item at front-end\n");
2060                scanf("%d",&item);
2061                first=insert_front(first,item);
2062                break;
2063            case 2:printf("enter the item at rear-end\n");
2064                scanf("%d",&item);
2065                first=insert_rear(first,item);
2066                break;
2067            case 3:printf("enter the item\n");
2068                scanf("%d",&item);
2069                printf("enter the position\n");
2070                scanf("%d",&pos);
2071                first=insert_pos(item,pos,first);
2072                break;
2073            case 4:first=delete_front(first);
2074                break;
2075            case 5:first=delete_rear(first);
2076                break;
2077            case 6:printf("enter the position\n");
2078                scanf("%d",&pos);
2079                first=delete_pos(pos,first);
2080                break;
2081            case 7:bubbleSort(first);
2082                printf("Items In Sorted Order are\n");
2083                display(first);
2084                break;
2085            case 8:printf("enter the no of nodes in 1\n");
2086                scanf("%d",&n);
2087                a=NULL;
2088                for(i=0;i<n;i++)
2089                {
2090                    printf("enter the item\n");
2091                    scanf("%d",&item);
2092                    a=insert_rear(a,item);
2093                }
2094                printf("enter the no of nodes in 2\n");
2095                scanf("%d",&n);
```

```
2095                    scanf("%d",&n);
2096                    b=NULL;
2097                    for(i=0;i<n;i++)
2098                    {
2099                        printf("enter the item\n");
2100                        scanf("%d",&item);
2101                        b=insert_rear(b,item);
2102                    }
2103                    a=concat(a,b);
2104                    printf("Concatenated list : \n");
2105                    display(a);
2106                    break;
2107                case 9:first=reverse(first);
2108                    printf("Reverse list : \n");
2109                    display(first);
2110                    break;
2111                case 10:printf("List : \n");
2112                    display(first);
2113                    break;
2114                case 11:exit(0);
2115                default:printf("Enter correct instruction!!!");
2116                    break;
2117            }
2118        }
2119        return 0;
2120    }
2121
```

```
 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
1
enter the item at front-end
1

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
2
enter the item at rear-end
2

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
2
enter the item at rear-end
3
```

```
"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"
 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
3
enter the item
4
enter the position
2

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
10
list :
1
4
2
3

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
4
```

```
"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"
item deleted at front end is-1

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
6
enter the position
2

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
10
list :
4
3

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
1
enter the item at front end
24
```

```
"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"
 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
7
Items in Sorted Order are
3
4
24

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
9
Reverse list :
24
4
3

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
8
enter the no of nodes in 1
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
enter the no of nodes in 1
2
enter the item
1
enter the item
2
enter the no of nodes in 2
2
enter the item
3
enter the item
4
Concatenated list :
1
2
3
4

 1:Insert_front
 2:Insert_rear
 3.Insert at specified position
 4:Delete_front
 5.Delete_rear
 6.Delete at specified position
 7.Sort
 8.Concatenate two lists
 9.Reverse the list
 10.Display_list
 11.Exit
enter the choice
11

Process returned 0 (0x0)   execution time : 77.834 s
Press any key to continue.
```