Experiment topic: **Design and Implementation of Audience Voting and Ranking System for TV Contest**

## 1. Experimental content

1. Description of the problem: In many TV competitions, usually after the performance of the contestants, the audience votes for the contestants through the buttons in their hands, and then counts the number of votes obtained by the contestants, and sorts them in descending order from high to low, so as to automatically generate champions , runner-up and runner-up. It is required to write an algorithm to simulate the function of the above system.
2. basic requirements
(1) First input the number of contestants (range: 1-9 ), and then use the malloc function to allocate the sequence table for storing player information according to the number of contestants;
(2) Store the player's number and name in the sequence table accordingly;
(3) The audience votes by pressing the buttons, press ' 1' to vote for No. 1 contestant, press ' 2' to vote for No. 2 contestant, and so on, and press ' 0' to mark the end of voting;
(4) Sorting after the voting is over. Various sorting algorithms are used here, and then the ranking is calculated for each player, and the rankings of the same votes are also the same;
(5) Design independently according to the requirements of the topic, and write a design report as required after the design is completed .

## 2. Original code:

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
#define OVERFLOW 0
#define ERROR 0
#define OK 1

//---------------------------------------------- ----------------------------------//
/*struct data
{
int number;
char name[20];
};
```

```
*/

typedef struct
{
    int key;
    int number;
    char name[20];
}RedType;
typedef struct
{
    RedType *r;
    int length;
}SqList;

int InitList(SqList &L);
int ListInsert(SqList &L,int i,RedType e,int MAXSIZE);
int modify(SqList &L,int b[10]); //votes are included in personal information
void BubbleSort(SqList &L); //bubble sort
void BInsertSort(SqList &L); //half insertion sort
int Partition(SqList &L,int low,int high); //quick sort
void QSort(SqList &L,int low,int high); //quick sort
void QuickSort(SqList &L); //Quick sort
void updown(SqList &L); //Replace in reverse order
void menu(); //function selection menu
void shuru(SqList &L); //function 1 input
void toupiao(SqList &L); //function 2 voting
void paixu(SqList &L); //Function 3 sorting
int show(SqList &L); //Function 4 display
void save(SqList &L); //function 5 save
//---------------------------------------------- ----------------------------------//

int InitList(SqList &L, int MAXSIZE)
{
    Lr=new RedType[MAXSIZE+1];
```

```
    if(!Lr) exit(OVERFLOW);
    L.length=0;
    return OK;
}

int ListInsert(SqList &L,int i,RedType e,int MAXSIZE)
{
    int j;
    if((i<1)||(i>L.length+1)) return ERROR;
    if(L.length==MAXSIZE) return ERROR;
    for(j=L.length;j>=i;j--)
        Lr[j+1]=Lr[j];
    Lr[i]=e;
    ++L.length;
    return OK;
}
//---------------------------------------------- ----------------------------------//Sorting functions

void BubbleSort(SqList &L) //bubble sort
{
    int m,flag,j;
    RedType t;
    m=L.length;flag=1;
    while((m>0)&&(flag==1))
    {
        flag=0;
        for(j=m;j>0;j--)
            if(Lr[j].key<Lr[j+1].key)
            {
                flag=1;
                t=Lr[j];
                Lr[j]=Lr[j+1];
                Lr[j+1]=t;
            }
```

```
                --m;
        }
}

void BInsertSort(SqList &L) //half insertion sort
{
        int i,low,high,m,j;
        for(i=2;i<=L.length;++i)
        {Lr[0]=Lr[i];
        low=1;high=i-1;
        while(low<=high)
        {
                m=(low+high)/2;
                if(Lr[0].key<Lr[m].key) high=m-1;
                else low=m+1;
        }
        for(j=i-1;j>=high+1;--j) Lr[j+1]=Lr[j];
        Lr[high+1]=Lr[0];
        }
}
int Partition(SqList &L,int low,int high) //quick sort
{
int pivotkey;
        Lr[0]=Lr[low];
        pivotkey=Lr[low].key;
        while(low<high)
        {
                while(low<high&&L.r[high].key>=pivotkey) --high;
                Lr[low]=Lr[high];
                while(low<high&&L.r[low].key<=pivotkey) ++low;
                Lr[high]=Lr[low];
        }
        Lr[low]=Lr[0];
        return low;
```

```
}

void QSort(SqList &L,int low,int high)
{
	int pivotloc;
	if(low<high)
	{
		pivotloc=Partition(L,low,high);
		QSort(L,low,pivotloc-1);
		QSort(L, pivotloc+1, high);
	}
}
void QuickSort(SqList &L)
{
	QSort(L,0,L. length);
}
void updown(SqList &L)
{
	int i;
	RedType e;
	for(i=1;i<=L.length/2;i++)
	{
		e=Lr[i];
		Lr[i]=Lr[L.length-i+1];
		Lr[L.length-i+1]=e;
	}

}
//----------------------------------------------	----------------------------------//Switch:case	subordinate characteristic operation function
void shuru(SqList &L)
{
	RedType e;
	int i,j;
```

```
    int MAXSIZE;
    printf("Please enter the total number of participants:");
    scanf("%d",&MAXSIZE);
    InitList(L,MAXSIZE);
    for(i=1;i<=MAXSIZE;i++)
    {
        printf("Please enter the player number:");
        scanf("%d",&j);
        e.number=j;
        printf("Please enter player name:");
        scanf("%s",&e.name);
        ListInsert(L,j,e,MAXSIZE);
    }
    printf("Enter successfully\n");
}

void toupiao(SqList &L)
{
    int vote[30];
    int b[10];
    memset(vote,0,sizeof(int)*30);
    memset(b,0,sizeof(int)*10);
    printf("Please enter the voting status (each number is separated by a space, and ends with
0)\n");
    int i=0,j=1;
    while(j!=0)
    {
        scanf("%d",&j);
        vote[i]=j;i++;
    }
    i=0;
    while(vote[i]!=0)
    {
        if(vote[i]==1) ++b[1];
```

```
            else if(vote[i]==2) b[2]++;
            else if(vote[i]==3) b[3]++;
            else if(vote[i]==4) b[4]++;
            else if(vote[i]==5) b[5]++;
            else if(vote[i]==6) b[6]++;
            else if(vote[i]==7) b[7]++;
            else if(vote[i]==8) b[8]++;
            else if(vote[i]==9) b[9]++;
            i++;
        }
        j=1;
        modify(L,b);
        printf("Enter successfully\n");
}
void paixu(SqList &L)
{
        int j;
        printf("1. Forward and backward bubble sort\n");
        printf("2. Half sort\n");
        printf("3. Quick sort\n");
        printf("Please select the sorting method you want to use:");
        scanf("%d",&j);
        if(j==1)
        {
            BubbleSort(L);
            printf("Sorting complete\n");
        }
        if(j==2)
        {
            BInsertSort(L);
            updown(L);
            printf("Sorting complete\n");
        }
        if(j==3)
```