“信息技术实践”课程报告

姓 名：

班 级：

学 号：

大连理工大学

Dalian University of Technology

**上机实践报告**

第1题：

设想网络上的一个发送者和64个接收者，发送者每秒有不超过128条的命令产生，每条命令包含一个512字节的头部command\_head\_t和至多4K字节的变长内容。command\_head\_t的结构如下：

typedef struct

{

int cmd\_no; //该命令的命令号，唯一识别一个命令

int version; //产生该命令的程序的版本

int detail\_len; //变产内容的实际长度

char \*content; //指向变长内容的指针

…

}command\_head\_t;

发送者根据命令号将这些命令分别发送给接收者去处理，例如：发送者产生c1，c2，c3，c4命令，并设定将c1，c2命令发送到接收者r1和r2，将c2、c3，c4命令发送到r3。

接收者执行接收到的命令，并相应修改自己的状态。

现在的问题是：在尽可能多的考虑各种可能的意外情况下（包括但不限于网络故障、传输错误、程序崩溃、停电…），如何设计命令的存储、发送、接收的流程，以保证命令的：

1)传输中的有序、无漏、无重复性

2)整个过程中命令和数据的正确性

3)多个同一类型的接收者（例如r1与r2）的状态可以在有限时间内趋于一致

最后，请针对你考虑到的意外情况，说明所采用的避免、解决或恢复方案。

代码：

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#define N 128

typedef struct

{

int cmd\_no;//该命令的命令号，唯一识别一个命令

int version;//产生该命令的程序的版本

int detail\_len;//变产内容的实际长度

char \*content;//指向变产内容的指针

}command\_head\_t;

command\_head\_t sender;//send command

command\_head\_t recipient[64];//64 recipient to receive command

char command\_content[]="Command sended.";

void function()

{

srand(time(0));

int i;

for (i=0;i<N;i++)

{

sender.cmd\_no=rand()%128+1;

sender.version=i;

sender.detail\_len=strlen(command\_content);

sender.content=command\_content;

recipient[rand()%63+0]=sender;

}

}

void print\_info()

{

int i;

printf("Recipient: \n");

for (i=0;i<64;i++)

{

printf("cmd\_no: %d\n", recipient[i].cmd\_no);

printf("version: %d\n", recipient[i].version);

printf("detail\_len: %d\n", recipient[i].detail\_len);

if (recipient[i].content)

{

printf("content: %s\n", recipient[i].content);

}

else

{

printf("content: No content. \n");

}

}

printf("\n");

}

int main()

{

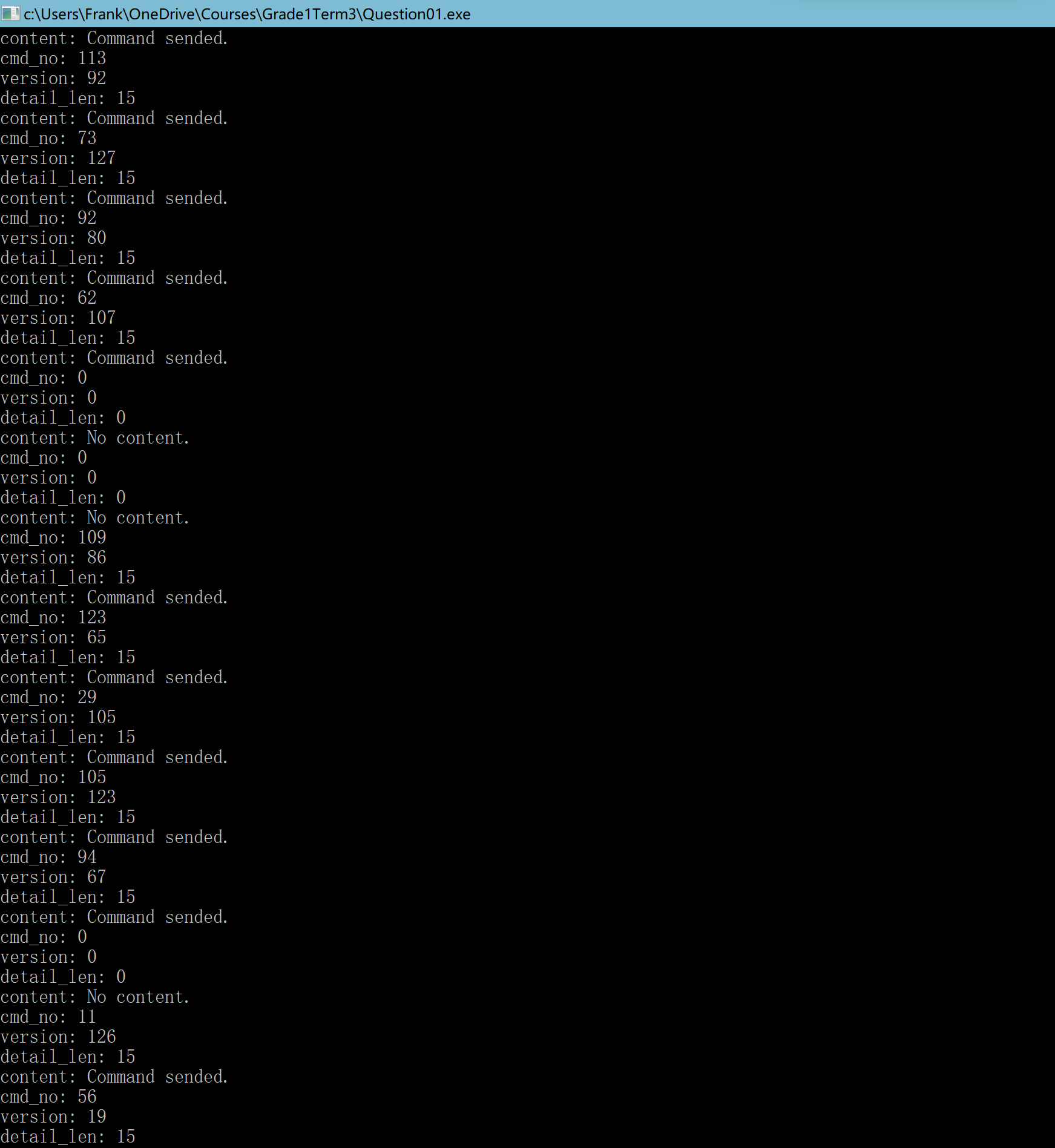
function();

print\_info();

system("pause");

return 0;

}



第2题：

一条1百万节点的单向链表，链表所有节点是按value字段从小到大的顺序链接；下面是一个节点的结构

typedef struct node\_t

{

int value; /\* 节点排序字段 \*/

int group; /\* 组号: 0,1,2,3,4,5,6,7,8,9 \*/

struct node\_t \*pnext; /\* 下个节点的指针 \*/

}node\_t;

node\_t head; /\*该单向链表的头节点，全局变量 \*/

试设计程序：针对各个group(0-->9)，根据value字段排序，输出各组top 10的节点。(top10是从小到大，取后面的大值top10.）

要求：尽量减少计算复杂度、遍历次数，不允许大量的辅助内存

代码：

//一条1百万节点的单向链表

//链表所有节点是按value字段从小到大的顺序链接

//试设计程序：针对各个group(0-->9)，根据value字段排序，输出各组top 10的节点。（top10是从小到大，取后面的大值top10.）

//要求：尽量减少计算复杂度、遍历次数，不允许大量的辅助内存

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define N 1000000

typedef struct node\_t

{

int value;

int group;//from 0~9

struct node\_t \*pnext;

}node\_t;

node\_t \*head;//head node of the list

node\_t \*initial\_list() //initialize the list(no bug)

{

printf("Start initializing the list...\n");

srand(time(0));

node\_t \*head\_temp=0,\*temp\_temp,\*tail\_temp=0;

int i;

for (i=0;i<N;i++)

{

temp\_temp=(node\_t \*)malloc(sizeof(node\_t));

temp\_temp->value=i;

temp\_temp->group=rand()%10+0;

temp\_temp->pnext=0;

if (!head\_temp)

{

head\_temp=tail\_temp=temp\_temp;

}

else

{

tail\_temp->pnext=temp\_temp;

tail\_temp=temp\_temp;

}

}

printf("List has been initialized. \n");

free(temp\_temp);

return head\_temp;

}

void traverse(node\_t \*Head)//(no bug)

{

int count=0;

int tab=0;//use tab to change the row while traversing and printing the list

while (Head)

{

printf(" group: %d ",Head->group);

printf(" value: %3d ",Head->value);

Head=Head->pnext;

count++;

tab++;

if (tab==3)

{

printf("\n");

tab=0;

}

if (count==N)

{

break;

}

}

printf("\n");

}

//sort by the 'group', and order by the 'value'.

//design a new struct for the array which is used to save the node\_t->value

typedef struct

{

int value;

}new\_node;

new\_node node\_array[10][10];

void print\_array(int n)

{

int i,j;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

if (n==0)

{

printf("Initial array:\n");

}

if (n==1)

{

printf("Array after bubblesort: \n");

}

for (i=0;i<10;i++)

{

for (j=0;j<10;j++)

{

printf("%d\t",node\_array[i][j].value);

}

printf("\n");

}

printf("\n");

}

void function(node\_t \*Head)

{

int i,j,m;

node\_t \*Temp=Head;

for (i=0;i<10;i++)

{

j=0;

while (Temp)

{

if (Temp->group==i)

{

node\_array[i][j].value=Temp->value;

j++;

if (j==10)

{

break;

}

}

Temp=Temp->pnext;

}

}

//traverse

print\_array(0);

//bubblesort on the node\_array

for (m=0;m<10;m++)

{

for (i=0;i<9;i++)

{

for (j=0;j<9-i;j++)

{

if (node\_array[m][j].value>node\_array[m][j+1].value)

{

new\_node exchange;

exchange=node\_array[m][j];

node\_array[m][j]=node\_array[m][j+1];

node\_array[m][j+1]=exchange;

}

}

}

}

//traverse

print\_array(1);

for (i=0;i<10;i++)

{

printf("The top10 value of %d group is: \n",i);

printf("value: %d\n",node\_array[i][9].value);

}

}

int main()

{

head=initial\_list();

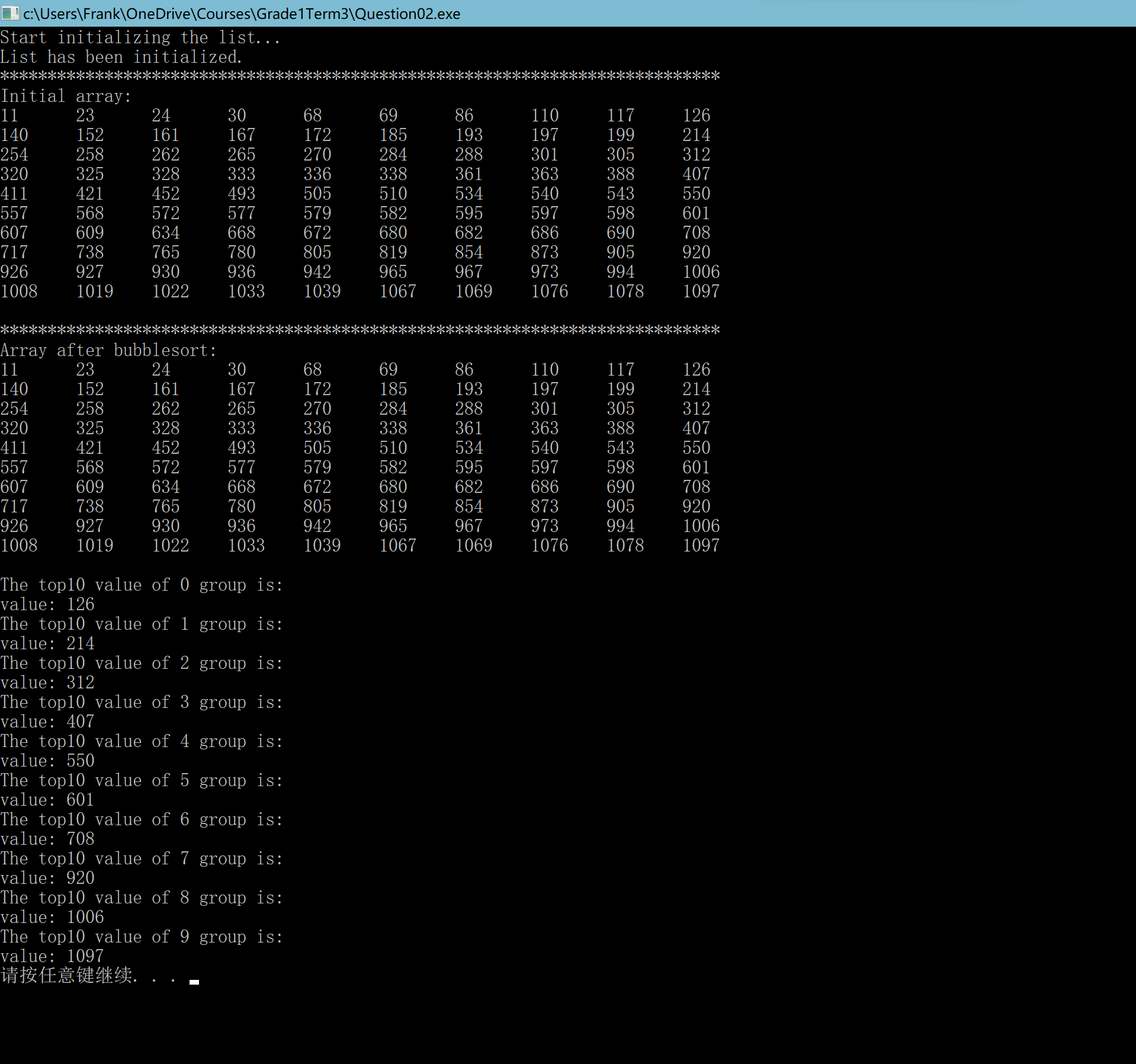
//traverse(head);

function(head);

system("pause");

return 0;

}



第3题：

青年歌手大赛记分程序，要求：

1.使用结构纪录选手的相关信息

2.使用链表或结构数组

3.对选手成绩进行排序并输出结果

4.利用文件纪录初赛结果，在复赛时将其从文件中读出程序，累加到复赛成绩中。

并将比赛最终结果写入文件中。

代码：

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#define N 10//set 10 competitors

struct Person

{

    int Id;

    double Score\_first;

    double Score\_final;

    double Sum;

};

struct Person competitor[N];//struct array

void initial\_first()

{

    srand(time(0));

    int i;

    for (i=0;i<N;i++)

    {

        competitor[i].Id=i+1;

        competitor[i].Score\_first=rand()%20+80;

    }

}

void bubblesort()

{

    int i,j;

    for (i=0;i<N-1;i++)

    {

        for (j=0;j<N-i-1;j++)

        {

            if (competitor[j].Score\_first<competitor[j+1].Score\_first)

            {

                struct Person temp;

                temp=competitor[j];

                competitor[j]=competitor[j+1];

                competitor[j+1]=temp;

            }

        }

    }

}

void file\_write\_First\_Score()

{

    FILE \*fp\_Id=fopen("C:/Users/Frank/OneDrive/CoursesGrade1Term3/filetemp/id.txt","w+");

    FILE \*fp\_Score=fopen("C:/Users/Frank/OneDrive/CoursesGrade1Term3/filetemp/first\_score.txt","w+");

    int i;

    for (i=0;i<N;i++)

    {

        fputc(competitor[i].Id,fp\_Id);

    }

    for (i=0;i<N;i++)

    {

        fputc(competitor[i].Score\_first,fp\_Score);

    }

    fclose(fp\_Id);

    fclose(fp\_Score);

}

void display\_byFirst()

{

    int i;

    printf("Output the first score from high to low: \n");

    for (i=0;i<N;i++)

    {

        printf("No.%d competitor ranks %d, final score is: %lf", competitor[i].Id, i+1, competitor[i].Score\_first);

        printf("\n");

    }

}

int First\_Score[N];

void file\_read\_First\_Score()

{

    FILE \*fp\_Score=fopen("C:/Users/Frank/OneDrive/CoursesGrade1Term3/filetemp/first\_score.txt","w+");

    int i;

    for (i=0;i<N;i++)

    {

        First\_Score[i]=fgetc(fp\_Score);

    }

    fclose(fp\_Score);

}

void initial\_final()

{

    srand(time(0));

    int i;

    for (i=0;i<N;i++)

    {

        competitor[i].Score\_final=rand()%30+70;

        competitor[i].Sum=First\_Score[i]+competitor[i].Score\_final;

    }

}

void selectsort()

{

    int i,j;

    for (i=0;i<N;i++)

    {

        for (j=i+1;j<N;j++)

        {

            if (competitor[j].Sum>competitor[i].Sum)

            {

                struct Person temp;

                temp=competitor[j];

                competitor[j]=competitor[i];

                competitor[i]=temp;

            }

        }

    }

}

void file\_write\_Final\_Score()

{

    FILE \*fp\_Id=fopen("C:/Users/Frank/OneDrive/CoursesGrade1Term3/filetemp/id.txt","w+");

    FILE \*fp\_Score=fopen("C:/Users/Frank/OneDrive/CoursesGrade1Term3/filetemp/final\_score.txt","w+");

    int i;

    for (i=0;i<N;i++)

    {

        fputc(competitor[i].Sum,fp\_Score);

    }

    fclose(fp\_Id);

    fclose(fp\_Score);

}

void display\_bySum()

{

    int i;

    printf("Output the final score from high to low: \n");

    for (i=0;i<N;i++)

    {

        printf("No.%d competitor ranks %d, final score is: %lf", competitor[i].Id, i+1, competitor[i].Sum);

        printf("\n");

    }

}

int main()

{

    initial\_first();

    bubblesort();

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FIRST SCORE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    file\_write\_First\_Score();

    display\_byFirst();

    initial\_final();

    file\_read\_First\_Score();

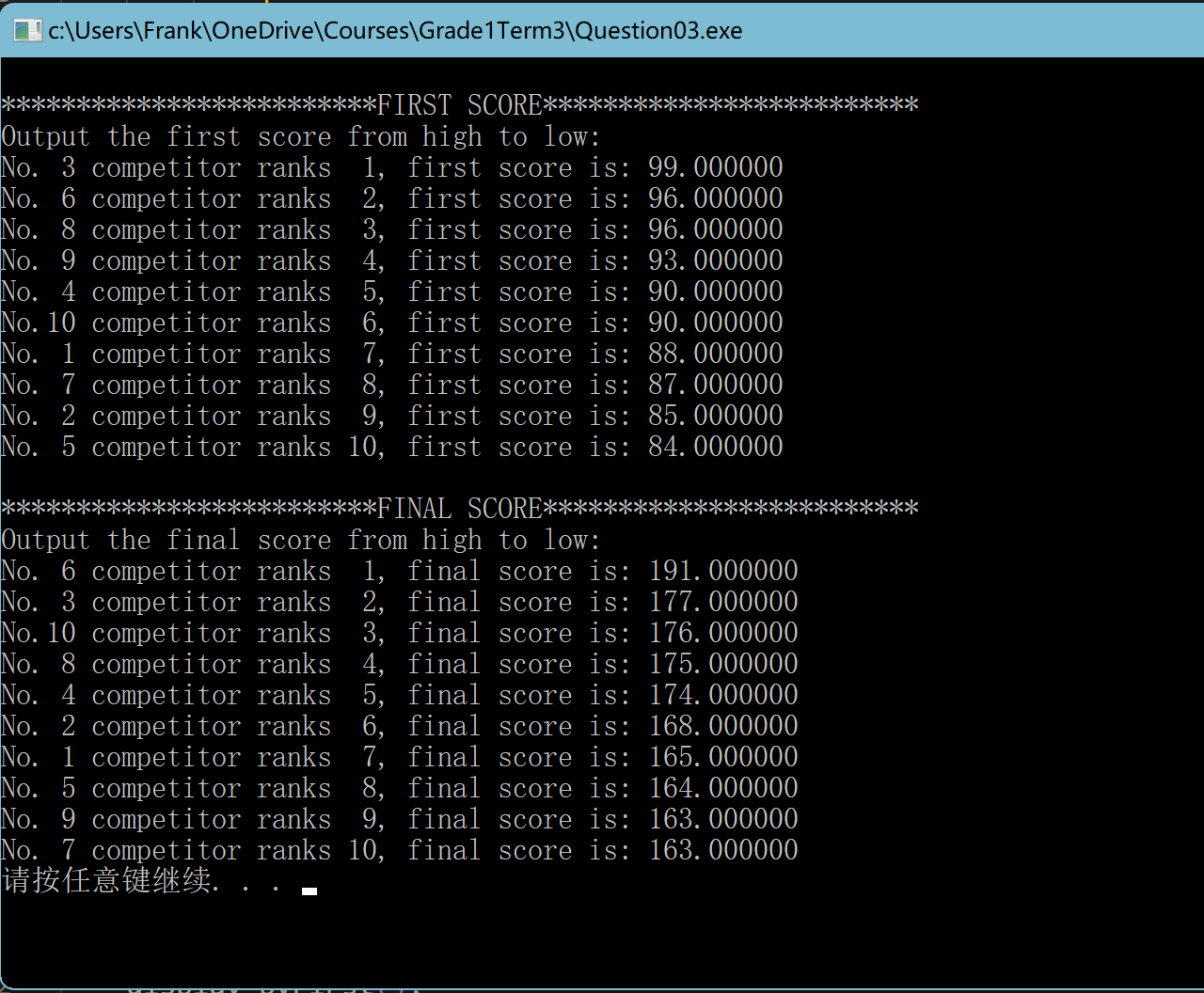
    selectsort();

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FINAL SCORE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    display\_bySum();

    file\_write\_Final\_Score();

    system("pause");

    return 0;}

第4题：

有5个字符串，首先将它们按照字符串中的字符个数由小到大排列，再分别取出每个字符串的第三个字母合并成一个新的字符串输出（若少于三个字符的输出空格）。要求：利用字符串指针和指针数组实现。

代码：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 5

char \*str1=(char \*)"China";

char \*str2=(char \*)"America";

char \*str3=(char \*)"Japan";

char \*str4=(char \*)"Russia";

char \*str5=(char \*)"England";

void function()

{

    char \*str\_array[N]={str1, str2, str3, str4, str5};

    int i,j;

    for (i=0;i<N-1;i++)

    {

        for (j=0;j<N-1-i;j++)

        {

            if (strlen(str\_array[j])>strlen(str\_array[j+1]))

            {

                char \*temp;

                temp=str\_array[j];

                str\_array[j]=str\_array[j+1];

                str\_array[j+1]=temp;

            }

        }

    }

    printf("Output the strings by their's length from short to long: \n");

    for (i=0;i<N;i++)

    {

        printf("%s\n",str\_array[i]);

    }

    char new\_array[N+1];

    for (i=0;i<N;i++)

    {

        char \*temp=str\_array[i];

        char temp\_char=temp[2];

        new\_array[i]=temp\_char;

    }

    new\_array[N]='\0';

    printf("Pick up the third letter from the sorted strings...\n");

    printf("The new string is: \n%s\n",new\_array);

}

int main()

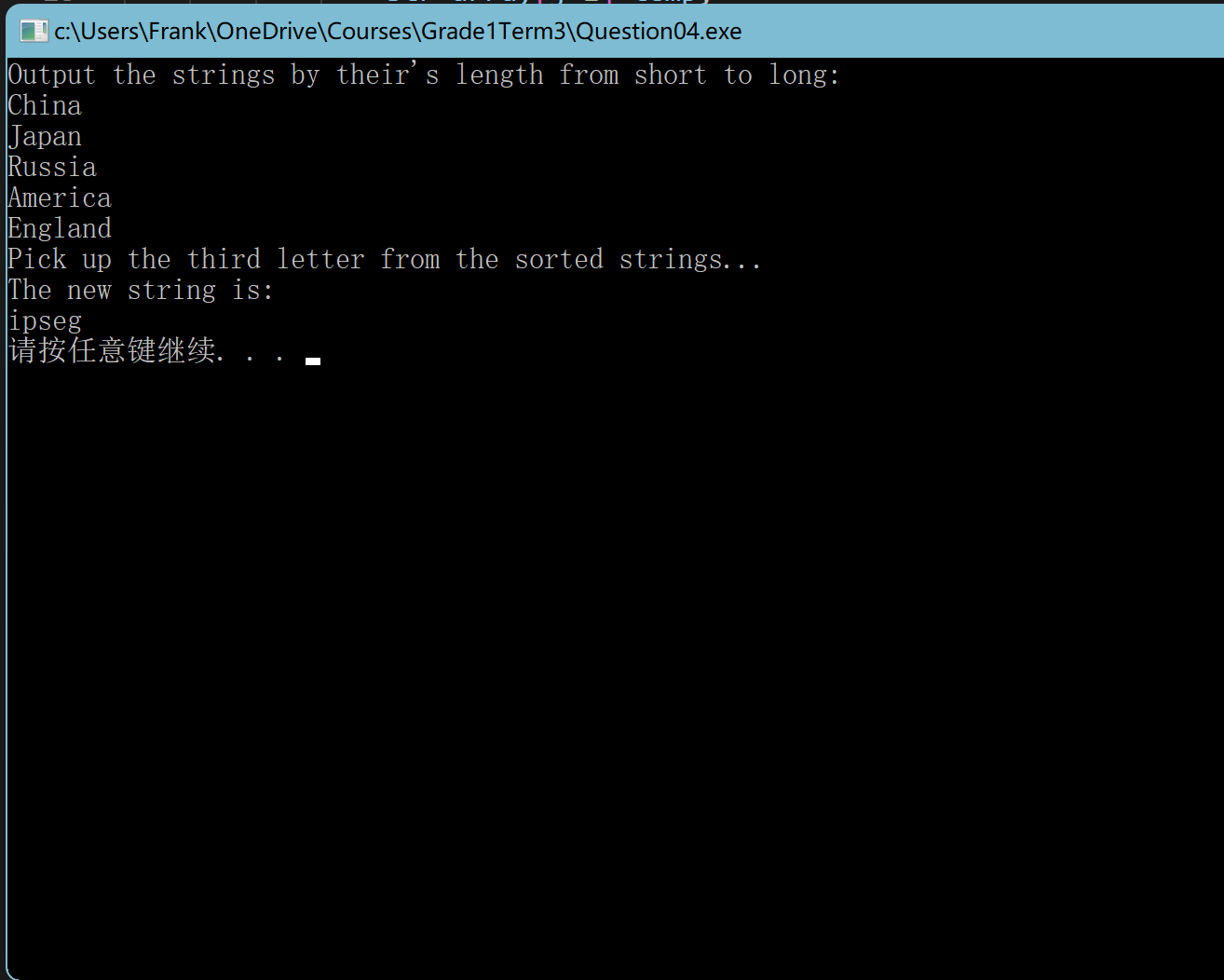
{

    function();

    system("pause");

    return 0;

}



第5题：

定义一个动态数组，长度为变量n，用随机数给数组各元素赋值，然后对数组各单元排序，定义swap函数交换数据单元，要求参数使用指针传递。

代码：

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define n 10

int \*dynamic\_array;

void initial\_array()

{

    dynamic\_array=(int\*)malloc(n\*sizeof(int));

    srand(time(0));

    int i;

    for (i=0;i<n;i++)

    {

        dynamic\_array[i]=rand()%100+0;

    }

}

void sort()

{

    int i,j;

    for (i=0;i<n-1;i++)

    {

        for (j=0;j<n-1-i;j++)

        {

            if (dynamic\_array[j]>dynamic\_array[j+1])

            {

                int temp;

                temp=dynamic\_array[j];

                dynamic\_array[j]=dynamic\_array[j+1];

                dynamic\_array[j+1]=temp;

            }

        }

    }

}

void swap(int \*x, int \*y)

{

    int temp;

    temp=\*x;

    \*x=\*y;

    \*y=temp;

}

void display()

{

    int i;

    printf("The dynamic array is: ");

    for (i=0;i<n;i++)

    {

        printf("%d ",dynamic\_array[i]);

    }

    printf("\n");

}

int main()

{

    initial\_array();

    sort();

    display();

    swap(&dynamic\_array[0],&dynamic\_array[1]);

    printf("Swap the first val and the second val: \n");

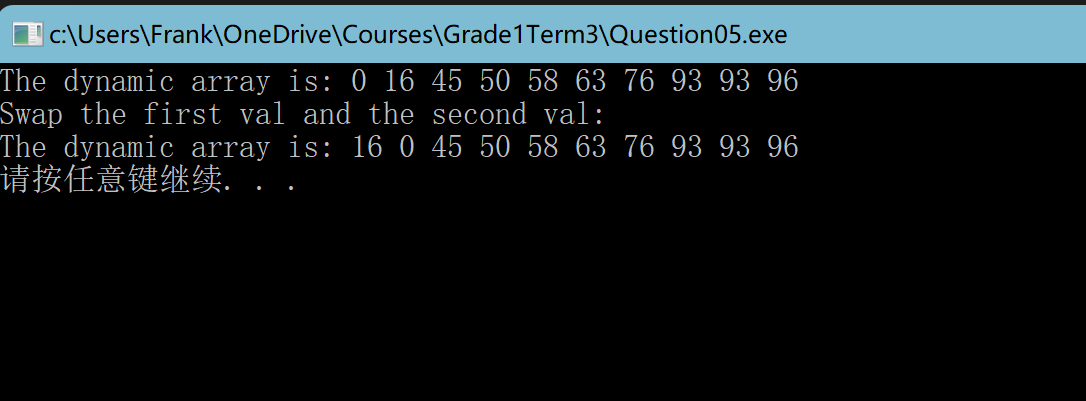
    display();

    free(dynamic\_array);

    system("pause");

    return 0;

}



第6题：

实现模拟彩票的程序设计：随机产生6个数字，与用户输入的数字进行比较，输它们相同的数字个数（使用动态内存分配）。

代码：

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define N 6

int \*lottery\_array;

int \*customer\_array;

void Lottery()

{

    srand(time(0));

    lottery\_array=(int\*)malloc(N\*sizeof(int));

    int i;

    for (i=0;i<N;i++)

    {

        lottery\_array[i]=rand()%9+0;

    }

}

void Customer()

{

    srand(time(0));

    customer\_array=(int\*)malloc(N\*sizeof(int));

    printf("Input %d numbers: \n",N);

    int i;

    for (i=0;i<N;i++)

    {

        scanf("%d",&customer\_array[i]);

    }

}

void Function()

{

    int count=0;

    int i,j;

    for (i=0;i<N;i++)

    {

        for (j=0;j<N;j++)

        {

            if (customer\_array[j]==lottery\_array[i])

            {

                count++;

            }

        }

    }

    printf("The lottery array is: \n");

    for (i=0;i<N;i++)

    {

        printf("%d ",lottery\_array[i]);

    }

    printf("\n");

    printf("The amount the same number between the lottery and the customer's input numbers is: %d\n",count);

    return;

}

int main()

{

    Lottery();

    Customer();

    Function();

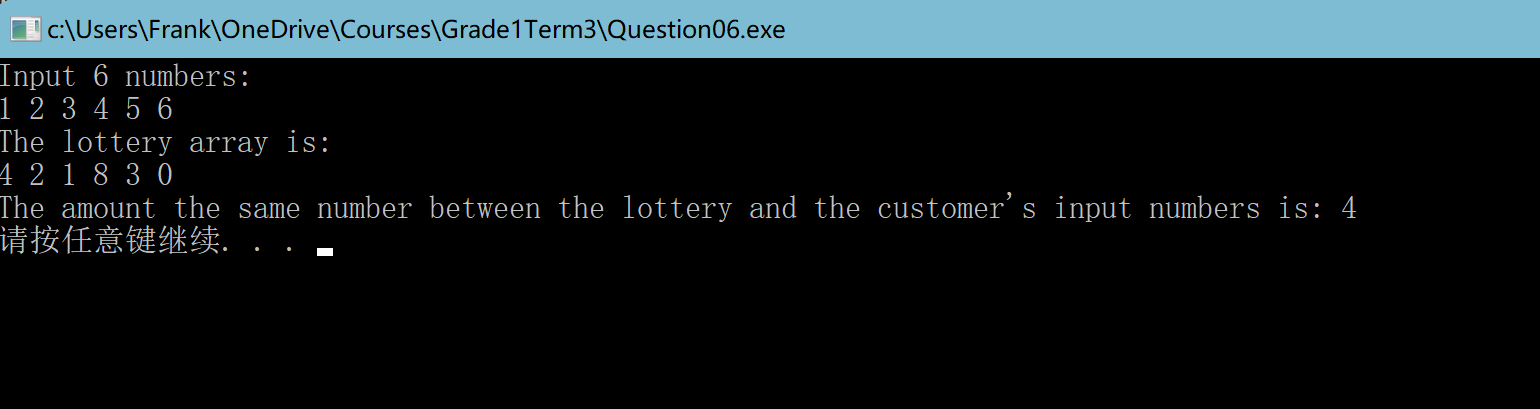
    free(lottery\_array);

    free(customer\_array);

    system("pause");

    return 0;

}



第7题：

加密程序:由键盘输入明文，通过加密程序转换成密文并输出到屏幕上。 算法：明文中的字母转换成其后的第4个字母,例如，A变成E(a变成e)，Z变成D,非字母字符不变；同时将密文每两个字符之间插入一个空格。例如，China转换成密文为G l m r e。要求：在函数change中完成字母转换，在函数insert中完成增加空格，用指针传递参数。

代码：

#include <stdio.h>

#include <stdlib.h>

#define N 100

void change(char \*plaintext)

{

int i;

for (i=0;plaintext[i]!='\0';i++)

{

if (plaintext[i]>='a' && plaintext[i]<='v')

{

plaintext[i]+=4;

}

else if (plaintext[i]>'A' && plaintext[i]<'V')

{

plaintext[i]+=4;

}

else if (plaintext[i]>='w' && plaintext[i]<='z')

{

plaintext[i]-=22;

}

else if (plaintext[i]>='W' && plaintext[i]<='Z')

{

plaintext[i]-=22;

}

else

{

continue;

}

}

}

void insert(char \*plaintext, char \*ciphertext)

{

int i,j;

for (i=0,j=0;plaintext[i]!='\0';i++)

{

ciphertext[j]=plaintext[i];

ciphertext[j+1]=' ';

j+=2;

}

ciphertext[j]='\0';

printf("The ciphertext is: ");

puts(ciphertext);

}

int main()

{

char plaintext[N];

char ciphertext[2\*N];

printf("Input the plaintext: ");

gets(plaintext);

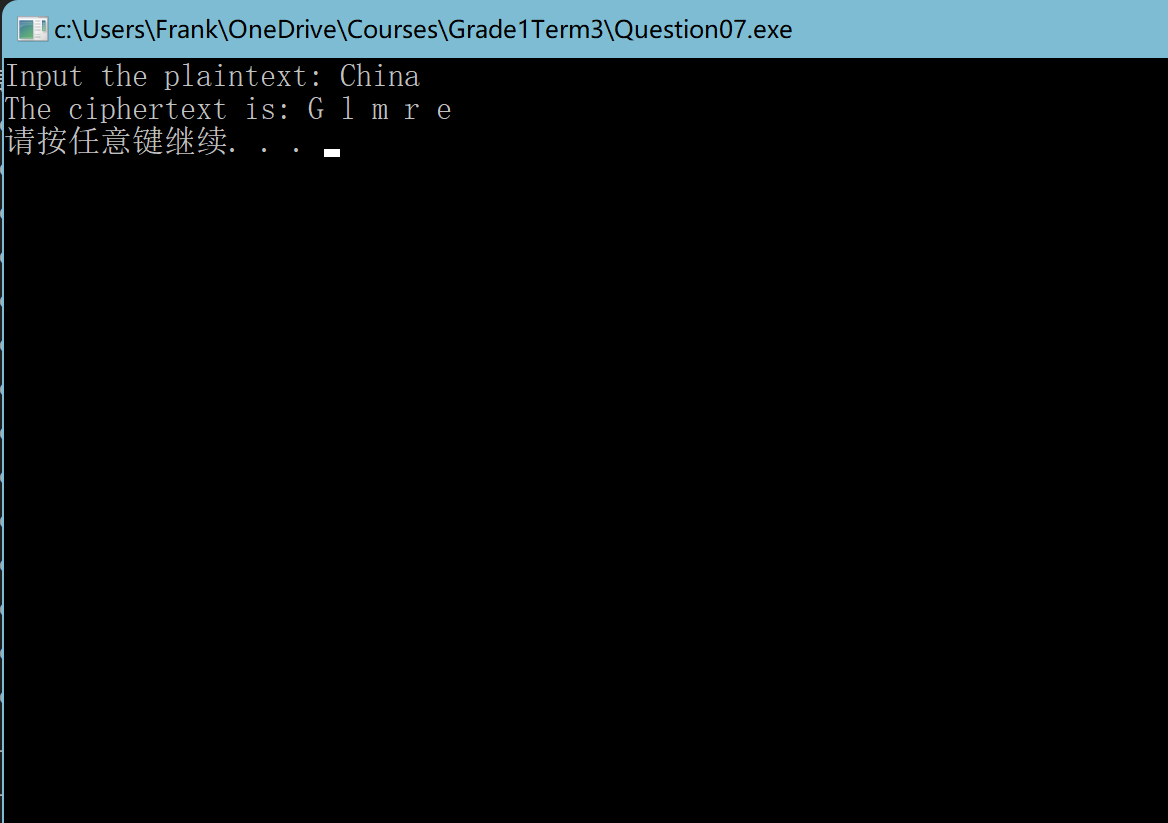
change(plaintext);

insert(plaintext,ciphertext);

system("pause");

return 0;

}



第8题：

用递归做汉诺塔游戏！

代码：

//递归做汉诺塔

#include <stdio.h>

#include <stdlib.h>

/\*

moving main function:

x,y,z poles. n disks.

1.Move n-1 disks from x to the y pole;

2.Move 1 disk to the z pole;

3.Move n-1 disks from y to z pole.

\*/

void moveDisk(int n, char x, char y, char z)

{

if(n==1)

{

printf("The %d disk: from %c to %c\n",n,x,z);

}

else

{

moveDisk(n-1,x,y,z);

printf("The %d disk: from %c to %c\n",n,x,z);

moveDisk(n-1,y,x,z);

}

}

int totalStep(int n)

{

if (n==1)

{

return 1;

}

else

{

return 2\*totalStep(n-1)+1;

}

}

int main()

{

char pole1 = 'x';

char pole2 = 'y';

char pole3 = 'z';

int n;

int step\_amount;

printf("Input the number of the disk: ");

scanf("%d",&n);

step\_amount=totalStep(n);

printf("You should go %d steps to get disks from x to z pole.\n",step\_amount);

printf("The each step is as follows: \n");

moveDisk(n,pole1,pole2,pole3);

system("pause");

return 0;

}

