第一题

#include <stdio.h>

/\*

f(x[n]移动到z) = f(x[n-1]移动到y) + f(y移动到z)

结束是n = 1

\*/

void Move(int n, char x, char z) //用于输出汉诺塔的移动轨迹，第n片，从x挪到z

{

printf("%d\t%c ----> %c \n", n, x, z);

}

void Hanoi(int n, char x, char y, char z) //汉诺塔移动

{

if (n == 1)

{

Move(n, x, z); //如果为最顶上一片，直接挪

}

else

{

Hanoi(n - 1, x, z, y); //首先要去挪x顶上的那片y，给x->z腾出空间

Move(n, x, z); //x->z

Hanoi(n - 1, y, x, z); //将刚才挪走的y再次挪回z（x->z的顶上）

}

}

int main()

{

Hanoi(3, 'X', 'Y', 'Z'); //进行第一次移动

}

第二题

#include<stdio.h>

#define ll long long

/\*

f(x) = a[x] \* f(x+1)

结束值，a[x] = 0

\*/

ll Multiply(long Arr[], int Index) //递归求元素的积，需保证数组最后以0结尾

{

if (Arr[Index] == 0) //如果遇到0了，说明需要乘的数以全部递归

{

return 1; //返回乘数1

}

return Arr[Index] \* Multiply(Arr,Index + 1); //向上一层返回自己乘以下一层

}

int main()

{

long Arr[] = { 1,2,3,4,5,6,7,8,9,88,0}; //例子

printf("Multiplied = %lld", Multiply(Arr, 0)); //输出

return 0;

}

第三题

#include <stdio.h>

/\*

f(x) = { f(x) (当f(x)大于f（x+1）)

{ f(x + 1) (当f（x+1）大于f(x))

\*/

static int IndexOfMax = 0;

int Max(int Arr[],int Index) //从index（min = 0）开始，获取最大值，数组需以0结尾

{

int Later = 0; //用于记录下一个Max的返回值

IndexOfMax = Index; //假设最大值就是自己

if (Arr[Index] == 0) {

return Arr[Index]; //如果到末尾了，就返回最大值是自己

}

else

{

Later = Max(Arr,Index + 1); //获取index+1之后的最大值

if (Arr[Index] > Later)

{

IndexOfMax = Index; //如果自己比 （（index+1）之后的最大值）大，那就标记自己

return Arr[Index]; //返回自己的值

}

else

{

return Later; //如果自己小，那么就返回index+1的最大值

}

}

}

int main()

{

int Arr[] = { 1,2,3,4,5,6,7,8,9,88,8,6,7,1,0 }; //例子

printf("Max = %d", Max(Arr, 0)); //输出

return 0;

}

第四题

#include <stdio.h>

/\*

Arr[I2C(Num / 10)] = Num % 10

\*/

int Int2Char(int Num, char Arr[]) //递归Num转Arr（不会自动补充\0）

{

int Pos = 0; //用于记录写入位置

if (Num == 0)

{

return 0; //如果Num == 0.已经是数字最高位，返回0

} //上一个数字应该对应写在Arr[0]

else

{

Pos = Int2Char(Num / 10, Arr); //获得自己是数字的第几位(从左到右)

Arr[Pos] = (Num % 10) + '0'; //将数字转化为字符

return Pos + 1; //返回下一位

}

}

int main()

{

int Num = 20040218;

char Arr[500] = {0}; //初始化为\0,自带结束符

Int2Char(Num, Arr); //转化

printf("%s", Arr);

return 0;

}

第五题

#include <stdio.h>

void InputArray(int Arr[], int Len) //输入长度为Len的数组，放入Arr

{

printf("请输入一个长度为 %d 的数组\n", Len);

for (int i = 0; i < Len; i++)

{

scanf\_s("%d", &Arr[i]);

}

}

void Insert(int Arr[], int &Len, int Adder) //向长度为Len数组Arr中插入Adder

{

int Pos = Len; //原数组从大到小，先假设加入数最小POS最后

for (int i = 0; i < Len;i++) //寻找找到可以插入的位置

{

if (Adder > Arr[i])

{

Pos = i; //记录位置

break;

}

}

for (int i = Len; i > Pos; i--) //将Pos到末尾的数，均向后移动一位

{

Arr[i] = Arr[i - 1];

}

Arr[Pos] = Adder; //写入移动产生的空间

Len++; //数组长度加一

}

void BubbleSort(int Arr[], int Len) //标准的冒泡排序法

{

int Temp = 0;

bool Flag = true;

for (int i = 0; i <= Len - 2 && Flag; i++)

{

Flag = false;

for (int j = 0; j <= Len - 2 - i;j++)

{

if (Arr[j] < Arr[j + 1])

{

Temp = Arr[j];

Arr[j] = Arr[j + 1];

Arr[j + 1] = Temp;

Flag = true;

}

}

}

}

void OutputArray(int Arr[], int Len) //输出数组

{

for (int i = 0; i < Len; i++)

{

printf("%d\t", Arr[i]); //循环输出

if ((i + 1) % 5 == 0)

{

printf("\n"); //每5个空一行

}

}

}

int main()

{

int Len = 10; //长度为10

int Arr[100]; //数组长度至少是Len+1

InputArray(Arr, Len); //输入

BubbleSort(Arr, Len); //排序

Insert(Arr, Len, 46); //插入

OutputArray(Arr, Len); //输出

return 0;

}

第六题

#include <stdio.h>

int CommonMutiple(int a, int b) //用于计算a和b的最小公倍数

{

int i = a;

for (; (i % b != 0); i += a) //以a为步长增加i循环，直到被b整除

{

}

return i; //返回这个数i

}

int main()

{

int Arr[5]; //初始化五个数

for (int i = 0; i < 5; i++) //输入

{

scanf\_s("%d", &Arr[i]);

}

for (int i = 0; i < 5 - 1;i++) //双重循环

{

for (int j = 0; j < 5 - 1 - i; j++) //分别对其中两个求公倍数

{ //直到就剩一个数字

Arr[j] = CommonMutiple(Arr[j], Arr[j + 1]); //就是五个数的公倍数

}

}

printf("%d", Arr[0]); //输出

return 0;

}

第七题

#include <stdio.h>

void TimerCounter1() //使用静态局部变量实现

{

static int Num1 = 0; //静态局部变量

Num1++;

printf("C1已被调用了%d次\n", Num1); //输出

}

int Num2 = 0; //全局变量

void TimerCounter2() //使用全局变量实现

{

Num2++;

printf("C2已被调用了%d次\n", Num2); //输出

}

int main()

{

for (int i = 0; i < 10; i++) //分别调用十次

{

TimerCounter1();

}

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

{

TimerCounter2();

}

return 0;

}

第八题

#include <stdio.h>

bool IsAlphabetOrNum(char Input) //用于判断一个char是不是字母或数字

{

if (Input >= '0' && Input <= '9') return true; //三种情况依次判断

if (Input >= 'a' && Input <= 'z') return true;

if (Input >= 'A' && Input <= 'Z') return true;

return false; //都不是

}

void TrimRecite(char Arr[], int Len)

{

char TempChar = 0; //用于缓存上一个字符

int i = 0, j = 0; //i表示读取的位置，j写入位置

for (; i < Len + 1; i++)

{

if (IsAlphabetOrNum(Arr[i]))

{

TempChar = 0; //如果当前符合要求，清空缓存

Arr[j] = Arr[i]; //对应位写入

j++; //写入位置调整为下一个

}

else

{ //如果不符合要求

if (TempChar != Arr[i]) //与缓存的不一致，认为是新的符号

{

TempChar = Arr[i]; //缓存符号

Arr[j] = Arr[i]; //写入这个符号

j++; //写入位置调整为下一个

}

else

{

//什么都不干 //如果是个符号，而且与缓存一致，直接忽略它

}

}

}

for (int i = j; i < Len; i++) //循环到最后一个j之后，剩下补\0

{

Arr[i] = '\0';

}

}

int main()

{

char Arr[500] = "a\*\*\*b--c+++d###$#%#$%e---+~~"; //测试用字符

int Len = 0;

for (; Arr[Len] != '\0'; Len++); //获取长度

TrimRecite(Arr, Len); //处理

printf("%s", Arr);

}

第九题

#include <stdio.h>

void JoinArr(int Arr1[], int Arr2[], int Len1, int Len2,int Arr3[]) //将数组2合并到1，二者均从大到小

{

int i = 0, j = 0, k=0; //分别指向 Arr1读取 Arr2读取 Arr3写入

for (;i < Len1 && j < Len2;k++) //循环到Arr1、Arr2至少一个完全排序读取

{

if (Arr1[i] >= Arr2[j]) //分别读取对应Arr1/2 进行排序

{

Arr3[k] = Arr1[i]; //选择大的写入

i++; //读取标记转为下一个

}

else

{

Arr3[k] = Arr2[j]; //同理

j++;

}

}

for (;i < Len1;i++,k++) //以下两个for有一个会因为i/j = len1/2 跳过

{ //另一个则将没读取完的数组全部写入

Arr3[k] = Arr1[i]; //（因为他们比较小，另一个数组读完了还没到）

}

for (;j < Len2;j++, k++)

{

Arr3[k] = Arr2[j];

}

}

int main()

{

int a[5] = { 10,8,6,4,2 }; //插入数组

int b[5] = { 9,7,5,3,1 };

int c[10]; //输出数组

JoinArr(a, b, 5, 5, c); //插入

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

{

printf("%d ", c[i]); //输出

}

}

第十题

#include <stdio.h>

#include <math.h>

#define ll long long

ll AlreadyFact[999] = {0}; //把现有的阶乘储存

ll Fact(int N) //计算阶乘

{

if (AlreadyFact[N] != 0) //如果查找的阶乘已有，则输出

{

return AlreadyFact[N];

}

if (N == 0) //0! = 1

{

AlreadyFact[N] = 1;

return 1;

}

AlreadyFact[N] = N \* Fact(N - 1); //递推公式

return AlreadyFact[N];

}

double Taylor(float X,int Index) //泰勒展开e^x

{

double Result = 0;

for (int i = 0; i < Index; i++)

{

Result += pow(X, i) / Fact(i); //利用公式算到index

}

return Result;

}

int main()

{

const double NumOfE = 2.7182818284590452353602874713526624977572;

//对e的值进行定义

for (float X = 0; X <= 3; X += 0.2)

{

double EX1 = pow(NumOfE, X); //直接产生e^x

double EX2 = Taylor(X, 30); //泰勒展开

printf("当X = %.2f时 \n E^X = \t %lf \n Taylor = %lf \n \n", X, EX1, EX2);

}

}

第十一题

#include <stdio.h>

#define N 10

#define M 3

void GetNext(bool Rings[], int &Pos) //在Pos的人离开，for循环找到下一个在的人

{

int i = Pos + 1;

for (; Rings[i % N] = false;i++) //=false，没离开

{

}

Pos = i;

}

int GoOut(bool Rings[],int Pos, int Count,int Absent) //约瑟夫环，N个人在线，当前位置，报数计数，还有几个人在

{

if (Absent == 0) return 0; //如果没人了，结束

if (Rings[Pos] == false) //如果对应pos有人

{

Count++; //计数增加

if (Count == M) //如果计数器等于需要的报数

{

Rings[Pos] = true;

printf("%d 号走了", Pos); //标记离开

GetNext(Rings, Pos); //得到补位的人

printf("，%d 号轮到了\n", Pos);

GoOut(Rings, (Pos + 1) % N,1,Absent - 1); //下一个人报数

}

else

{

GoOut(Rings, (Pos + 1) % N, Count, Absent);//如果当前的人不在了，看下一个人

}

}

}

int main()

{

bool Rings[N] = {0}; //用于标记是否离开，0在，1离开

GoOut(Rings,0, 0, N); //约瑟夫环

return 0;

}

第十二题

#include <stdio.h>

int IntToStr(int Num, char Arr[]) //递归Num转Arr（不会自动补充\0）

{

int Pos = 0; //用于记录写入位置

if (Num == 0)

{

return 0; //如果Num == 0.已经是数字最高位，返回0

} //上一个数字应该对应写在Arr[0]

else

{

Pos = IntToStr(Num / 10, Arr); //获得自己是数字的第几位(从左到右)

Arr[Pos] = (Num % 10) + '0'; //将数字转化为字符

return Pos + 1; //返回下一位

}

}

int Differ(char Str1[], char Str2[], int Len) //返回汉明距离

{

int Count = 0; //用于计数

for (int i = 0; i < Len; i++)

{

if (Str1[i] != Str2[i])

{

Count++; //遍历，如果对应字符不同

}

}

return Count;

}

int main()

{

printf("请输入两个正整数\n"); //输入

int Num1, Num2;

scanf\_s("%d%d", &Num1, &Num2);

char Str1[100] = { 0 }, Str2[100] = { 0 };

int Len1 = IntToStr(Num1, Str1);

int Len2 = IntToStr(Num2, Str2); //转化

printf("长度为 %d 和 %d\n", Len1, Len2); //输出

printf("汉明距为 %d \n", Differ(Str1,Str2,Len1));

}