**第1关：百钱百鸡问题**

**本关任务：用蛮力法编程解决百钱百鸡问题。**

**百钱百鸡问题：中国古代数学家张丘建在他的《算经》中提出了著名的“百钱百鸡问题”：鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一；百钱买百鸡，翁、母、雏各几何?**

**#include <stdio.h>**

**void  main()**

**{**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**int x,y,z;**

**for(x=0;x<=20;x++)**

**{**

**for(y=1;y<=34;y++)**

**{**

**for(z=1;z<=100;z++)**

**{**

**if((100==x+y+z)&&(100==5\*x+3\*y+z/3.0))**

**{**

**printf("公鸡有%d只,母鸡有%d只,小鸡有%d只\n",x,y,z);**

**}**

**}**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**}**

**第2关：数字迷**

**本关任务：根据下面计算，用蛮力法解决数字迷问题，求出由 A、B、C 替代的相乘数字。**

**A B C A B**

**× A**

**-------------------**

**D D D D D D**

**#include <stdio.h>**

**void main()**

**{**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**int A,B,C,D;**

**long E,F;**

**for(A=3;A<=9;A++)//乘积是6位数，A必须大于2**

**{**

**for(D=1;D<=9;D++)**

**{**

**E=D\*100000+D\*10000+D\*1000+D\*100+D\*10+D;**

**if(E%A==0)**

**{**

**F=E/A;**

**if((F/10000==A)&&((F%100)/10==A))**

**{**

**if((F/1000)%10==F%10)**

**{**

**printf("%d\*%d=%d",F,A,E);**

**}**

**}**

**}**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**}**

**第3关：求3个数的最小公倍数**

**本关任务：输入 3 个数 a、b、c， 求这 3 个数的最小公倍数。**

**测试输入：2 4 6**

**预期输出：2,4,和6的最小公倍数是12**

**#include <stdio.h>**

**void main()**

**{**

**/\*\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**int x1,x2,x3,i;**

**scanf("%d %d %d",&x1,&x2,&x3);**

**i=1;**

**while(1){**

**if((i%x1==0)&&(i%x2==0)&&(i%x3==0))**

**break;**

**i++;**

**}**

**printf("%d,%d,和%d的最小公倍数是%d",x1,x2,x3,i);**

**/\*\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**}**

**第4关：韩信点兵问题**

**本关任务：求韩信一共有多少兵。**

**韩信有一队兵，他想知道有多少人，便让士兵排队报数。**

**按从 1 至5报数，最末一个士兵报的数为 1；**

**按从 1 至 6 报数，最末一个士兵报的数为 5；**

**按从 1 至 7 报数，最末一个士兵报的数为 5；**

**按从 1 至 11 报数，最末一个士兵报的数为 10；**

**按从 1 至 13 报数，最末一个士兵报的数为 11。**

**请问韩信至少有多少兵？**

**#include <stdio.h>**

**void main()**

**{**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**for(int n=5;n<=9999;n++){**

**if(((n-1)%5==0)&&((n-5)%6==0)&&((n-5)%7==0)&&((n-10)%11==0)&&((n-11)%13==0))**

**printf("士兵最少有%d个",n);**

**}**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**}**

**第5关：白帽子红帽子问题**

**本关任务：正确输出他们各自所戴帽子的颜色。**

**厅内有 5 个人，他们均戴着帽子，有白帽子和红帽子。**

**已知戴白帽子的说真话，戴红帽子的说假话，请从他们各自提供的线索辨别谁戴白帽子，谁戴红帽子。**

**甲：我看见一个戴白帽子的**

**乙：我没有看见戴红帽子的**

**丙：我看见一个戴白帽子的，但不是甲**

**丁：我没有看见戴白帽子的**

**戊：我的帽子和丙一样**

**#include <stdio.h>**

**void main()**

**{**

**int c1,c2,c3,c4,c5;**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**for(int a=0;a<=1;a++)**

**for(int b=0;b<=1;b++)**

**for(int c=0;c<=1;c++)**

**for(int d=0;d<=1;d++)**

**for(int e=0;e<=1;e++)**

**{**

**c1=((b+c+d+e)==1)==a;**

**c2=((a+c+d+e)==4)==b;**

**c3=((b+d+e)==1)==c;**

**c4=((a+b+c+e)==0)==d;**

**c5=(c==e)==e;**

**if(c1&&c2&&c3&&c4&&c5)**

**{**

**if(a==1)**

**printf("甲戴帽子\n");**

**else**

**printf("甲戴红帽子\n");**

**if(b==1)**

**printf("乙戴白帽子\n");**

**else**

**printf("乙戴红帽子\n");**

**if(c==1)**

**printf("丙戴白帽子\n");**

**else**

**printf("丙戴红帽子\n");**

**if(d==1)**

**printf("丁戴白帽子\n");**

**else**

**printf("丁戴红帽子\n");**

**if(e==1)**

**printf("戊戴白帽子\n");**

**else**

**printf("戊戴红帽子\n");**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**}**

**第1关：计算多项式的值**

**本关任务：设计一个蛮力算法，对于给定的x0​，计算下面多项式的值：**

**P(x)=an​xn​+an−1​xn−1​+…+a1​x+a0​ 并确定该算法的最差效率类型。**

**已知多项式每一项的系数和变量x的值，可依次计算多项式每一项的值，并进行求和。**

**列表A中存放的是多项式的系数，A[i]表示ai​**

**测试输入： x=2  
A=[3,2,1,1] 预期输出： 19**

**//设计一个蛮力算法，对于给定的x0，计算下面多项式的值:**

**//    P(x)=anxn+an-1xn-1+…+a1x+a0**

**#include <stdio.h>**

**int evaluate(int x, int A[],int n) {**

**int p=A[0];//多项式的值**

**int temp;**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**for(int i=1;i<=n;i++){**

**temp=1;**

**for(int j=1;j<=i;j++){**

**temp=x\*temp;**

**}**

**p+=(A[i]\*temp);**

**}**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**return p;**

**}**

**void main(){**

**int x=2;**

**int A[]={3,2,1,1};**

**int n=3;**

**printf("%d",evaluate(x,A,n));**

**}**

**第2关：选择排序**

**本关提到的选择排序指的是简单选择排序，具体来说，对于一个长度为n的待排序序列（第1个元素在序列中的下标为0），要进行n-1趟排序：第i趟：在第i-1个元素到第n-1个元素中找到最小值，如果该最小元素不是第i-1个，则与第i-1个元素交换位置。**

**测试输入： A=['E','X','A','M','P','L','E']**

**预期输出：**

**['A', 'X', 'E', 'M', 'P', 'L', 'E']**

**['A', 'E', 'X', 'M', 'P', 'L', 'E']**

**['A', 'E', 'E', 'M', 'P', 'L', 'X']**

**['A', 'E', 'E', 'L', 'P', 'M', 'X']**

**['A', 'E', 'E', 'L', 'M', 'P', 'X']**

**['A', 'E', 'E', 'L', 'M', 'P', 'X']**

**//应用选择排序将序列E，X，A，M，P，L，E按照字母顺序排序。**

**#include <stdio.h>**

**void selectSort(char A[], int n) {**

**for (int i = 0; i < n-1; i++) {**

**int min = i;**

**for (int j = i+1; j < n; j++) {**

**if (A[j] < A[min])**

**min = j;**

**}**

**if (min != i) {**

**char temp = A[i];**

**A[i] = A[min];**

**A[min] = temp;**

**}**

**printf("[");**

**for (int k = 0; k < n-1; k++) {**

**printf("'%c', ", A[k]);**

**}**

**printf("'%c']", A[n-1]);**

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

**}**

**}**

**int main() {**

**char A[] = {'E', 'X', 'A', 'M', 'P', 'L', 'E'};**

**int n = sizeof(A) / sizeof(A[0]);**

**selectSort(A, n);**

**return 0;**

**}**

**第3关：冒泡排序**

**本关任务：应用冒泡排序将序列E，X，A，M，P，L，E按照字母顺序排序。**

**对于一个长度为n的待排序序列（第1个元素的下标为0），要进行n-1趟排序。 第1趟：从第0个到第n-1个元素，每相邻两个元素进行比较，如果第k个元素大于第k+1个元素，则进行交换；第i趟：从第0个到第n-i个元素，每相邻两个元素进行比较，如果第k个元素大于第k+1个元素，则进行交换。**

**应用冒泡排序将序列E，X，A，M，P，L，E按照字母顺序排序，要求输出每一趟排序的结果。 其中：列表A存放的是待排序序列。**

**测试说明**

**平台会对你编写的代码进行测试：**

**测试输入： A=['E','X','A','M','P','L','E']**

**预期输出：**

**['E', 'A', 'M', 'P', 'L', 'E', 'X']**

**['A', 'E', 'M', 'L', 'E', 'P', 'X']**

**['A', 'E', 'L', 'E', 'M', 'P', 'X']**

**['A', 'E', 'E', 'L', 'M', 'P', 'X']**

**['A', 'E', 'E', 'L', 'M', 'P', 'X']**

**['A', 'E', 'E', 'L', 'M', 'P', 'X']**

**#include <stdio.h>**

**void bubbleSort(char A[], int n) {**

**for (int i = 0; i < n-1; i++) {**

**for (int j = 0; j < n-i-1; j++) {**

**if (A[j] > A[j+1]) {**

**char temp = A[j];**

**A[j] = A[j+1];**

**A[j+1] = temp;**

**}**

**}**

**printf("[");**

**for (int k = 0; k < n-1; k++) {**

**printf("'%c', ", A[k]);**

**}**

**printf("'%c']", A[n-1]);**

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

**}**

**}**

**int main() {**

**char A[] = {'E', 'X', 'A', 'M', 'P', 'L', 'E'};**

**int n = sizeof(A) / sizeof(A[0]);**

**bubbleSort(A, n);**

**return 0;**

**}**

**第4关：蛮力法子串数量计算**

**本关任务： 有这样一个问题，在一段给定的文本中查找以A开始、以B结尾的子串的数量（例如，在CABAAXBYA中有4个这样的子串）。 a.为该问题设计一个蛮力算法并确定它的效率类型 b.为该问题设计一个更高效的算法**

**方法一： 对于一个长度为n的字符串（第1个字符的下标为0），依次检查字符串的第0到第n-2个字符，如果第i位置的字符是A，则从第i+1位置开始直到第n-1位置，扫描存在多少个字符B，假设存在k个字符B，则以第i位置的字符A开头，以字符B结束的子串有k个，重复这一过程直到字符串中第n-2位置的字符被检查完毕为止（不需要检查第n-1位置的字符是否是A，因为这是整个字符串的最后一个字符，后面不会再有字符B了）。 方法二： 上述方法是在确定了一个字符A之后，检查在它之后有多少个B，则可以确定以这个字符A开头，以字符B结束的子串个数。 我们也可以换一个思路，当确定一个字符B之后，如果已经记录了在它之前存在多少个字符A，则可以确定以该字符B结束，以字符A开头的子串个数。利用这一方式，只需要对给定字符串进行一趟遍历就可以统计出全部以A开头，以B结束的子串个数，相较第一种方法，算法效率得以提高。**

**测试输入： A='CABAAXBYA' 预期输出： 4 4**

**//有这样一个问题，在一段给定的文本中查找以A开始、以B结尾的子串的数量。**

**//（例如，在CABAAXBYA中有4个这样的子串）。**

**//   a.为该问题设计一个蛮力算法并确定它的效率类型**

**//    b.为该问题设计一个更高效的算法**

**#include <stdio.h>**

**int subString(char A[]){**

**int count=0;**

**int i, j;**

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

**{**

**if(A[i]=='A')**

**{**

**for(j=i+1; A[j]!='\0'; j++)**

**{**

**if(A[j]=='B')**

**{**

**count++;**

**}**

**}**

**}**

**}**

**return count;**

**}**

**int subStringPlus(char A[])**

**{**

**int count=0;**

**int i;**

**int numA=0;**

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

**{**

**if(A[i]=='A')**

**{**

**numA++;**

**}**

**else if(A[i]=='B')**

**{**

**count+=numA;**

**}**

**}**

**return count;**

**}**

**int main(){**

**char A[]="CABAAXBYA";**

**printf("%d\n",subString(A));**

**printf("%d",subStringPlus(A));**

**return 0;**

**}**

**第5关：邮局位置定位**

**本关任务： 假设x1​<x2​<x3​……<xn​是实数，它们分别代表坐落在一条直路上的n个村庄。我们需要在其中建一所邮局。 a.设计一个高效算法，求邮局所在位置，使得各村庄和该邮局的平均距离最小。 b.设计一个高效算法，求邮局所在位置，使得各村庄和该邮局之间的最大距离最小。**

**测试输入： A=[1,2,3,6,7,8.5,14,15]**

**预期输出： Position To minimize the average distance: 5 Position Make the maximum distance minimum: 6**

**测试输入： A=[1,2,3,4,7]**

**预期输出： Position To minimize the average distance: 3 Position Make the maximum distance minimum: 4**

**//假设x1<x2<x3…..<xn是实数，它们分别代表坐落在一条直路上的n个村庄。我们需要在其中建一所邮局。**

**//a.设计一个高效算法，求邮局所在位置，使得各村庄和该邮局的平均距离最小。**

**//b.设计一个高效算法，求邮局所在位置，使得各村庄和该邮局之间的最大距离最小。**

**#include <stdio.h>**

**#include <math.h>**

**#include <stdlib.h>**

**#include <string.h>**

**int positionA(float A[],int n){**

**int pos;**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**if(n%2==0)**

**{**

**pos=(n/2)+1;**

**};**

**if(n%2!=0)**

**{**

**pos=((n+1)/2);**

**};**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**return pos;**

**}**

**int positionB(float A[],int n){**

**int pos;**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**if(n%2==0)**

**{**

**pos=(n/2)+2;**

**};**

**if(n%2!=0)**

**{**

**pos=((n+1)/2)+1;**

**};**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**return pos;**

**}**

**//A=[1,2,3,4,7]**

**//A=[1,2,3,6,7,8.5,14,15]**

**void main(){**

**char row[1000];**

**fgets(row, 1000, stdin);**

**char \*token;**

**float A[1000];**

**int n = 0;**

**token = strtok(row, ",");**

**while (token != NULL) {**

**A[n++] = (float)atof(token);**

**token = strtok(NULL, ",");**

**}**

**int posA = positionA(A, n);**

**int posB = positionB(A, n);**

**printf("Position To minimize the average distance: %d\n", posA);**

**printf("Position Make the maximum distance minimum: %d", posB);**

**}**

**第6关：划分问题**

**本关任务： 考虑划分问题：给定n个正整数，把它们划分为元素之和相同但不想交的两个子集。（该问题并不总是有解），为该问题设计一个穷举查找算法，尽量减少该算法生成的子集数。**

**测试输入： A=[1,2,3,4] 预期输出： [2, 3]**

**//考虑划分问题：给定n个正整数，把它们划分为元素之和相同但不想交的两个子集。**

**//（该问题并不总是有解），为该问题设计一个穷举查找算法，尽量减少该算法生成的子集数。**

**#include <stdio.h>**

**#include <stdlib.h>**

**int partition(int\* items, int n) {**

**int sum = 0;**

**for (int i = 0; i < n; i++) {**

**sum += items[i];**

**}**

**if (sum % 2 == 1) {**

**return 0;**

**} else {**

**int mid = sum / 2;**

**if (items[n-1] > mid) {**

**return 0;**

**} else if (items[n-1] == mid) {**

**return (int[]){items[n-1]};**

**} else {**

**/\*\*\*\*\*\*\*\*\*\*  Begin  \*\*\*\*\*\*\*\*\*\*/**

**for (int j=0;j<mid;j++)**

**{**

**if(items[j]=mid-items[n-1])**

**return (int[]){items[n-1]};**

**return (int[]){items[j]}**

**}**

**/\*\*\*\*\*\*\*\*\*\*  End  \*\*\*\*\*\*\*\*\*\*/**

**}**

**}**

**}**

**int main() {**

**int A[] = {1, 2, 3, 4};**

**int n = sizeof(A) / sizeof(A[0]),m;**

**int\* result = partition(A, n);**

**if (result == NULL) {**

**printf("No subset with sum equal to half of total sum\n");**

**} else {**

**m = sizeof(result) / sizeof(result[0]);**

**printf("[");**

**for (int i = 0; i < m; i++) {**

**if(i==(m-1)){**

**printf("%d", result[i]);**

**} else{**

**printf("%d, ", result[i]);**

**}**

**}**

**printf("]");**

**free(result);**

**}**

**return 0;**

**}**