6-4 字符串重组 分数 100

全屏浏览题目 切换布局 作者 scs 单位 北京邮电大学

请写一个函数，实现按规则将两个字符串组和成一个字符串。

输入:
共2行，每行一个字符串，其中第一个字符串只包含大写字母，第二个字符串只包含数字且长度不超过15。
测试用例保证组合后的字符串长度不超过127。
输出:
只有一行，为组合后的字符串。

函数接口定义:

```
1 void recombination( char  str1[], char  str2[] ) ;
```

其中 str1 和 str2 都是用户传入的参数，为保存两个字符串的数组，且str2中仅包含数字。函数没有返回值。该函数的功能是将 str1 和 str2组合成一个字符串并将组合后的字符串保存在 str1 中。组合的规则为依次取str2中的字符插入到str1中。所有的字符插入的规则都是根据其数字和位置来决定。其中 str2中第一个字符插入的位置就是该数字所代表的位置，比如如果是0，就插在 str1的最前边，如果是1，就插在 str1中第一个字符的后边，依此类推。str2中其他字符插入的规则类似，只是每次都是从前边刚刚插入的位置开始数，比如刚插入的位置为10，当前待插入字符为0时则插在11，为1时则插在12，依此类推。测试用例保证插入过程中不会出现str1中最后一个字符的位置与待插入字符应该插入的位置之间有空位置的情况。

裁判测试程序样例

```
void recombination( char str1[] , char str2[] ) ;

int main()
{
    char    str1[LENA] , str2[LENB] ;

    scanf( "%s%s" , str1 , str2 ) ;
    recombination( str1 , str2 ) ;
    printf( "%s\n" , str1 ) ;

    return 0;
}
#include<string.h>
void recombination( char str1[] , char str2[] ){//str2 不再是个数组了，是指针，丢失数组长度信息，退化，不能用 sizeof(str2)，int*为 8
    // warning: ‘sizeof’ on array function parameter ‘str2’ will return size of ‘char*’ [-Wsizeof-array-argument]
//sizeof 吞数组
    //也不可以用 LENB，因为 a.cpp: In function ‘int main()’:
//a.cpp:12:10: warning: ignoring return value of ‘int scanf(const char*, ...)’, declared with attribute warn_unused_result [-Wunused-result]
     //scanf( "%s%s" , str1 , str2 ) ;
    // ~~~~~^~~~~~~~~~~~~~~~~~~~~~
    //因为 str2【】后面可能出现很大很离谱的数，造成数组越界问题
```

```
    //strlen 不含\0
```

总的思想是走移插，移动是移动数组省下位置，走是数的意思，数是逐渐增加的过程，是 b 完成的操作

```
  nt b=-1;

for(int i=0;i<strlen(str2);i++){
    char str3[LENA];
    b+=str2[i]-'0';

    for(int t=0;t<128;t++){
        str3[t]=str1[t];
    }

    for(int a=b+1+i;a<127;a++){
   str1[a+1]=str3[a];
   }
   str1[b+1+i]=str2[i];//加的 b，是加了多少走的数


}

   }
```

7-1 玩游戏一 分数 100

全屏浏览题目 切换布局 作者 scs 单位 北京邮电大学

你正在玩一款新的游戏，在游戏中你有N个用于给你的战士补充能量的道具和M个战士。这N个道具都有一个能量值，代表该道具能给战士提供的总能量，给战士补充后该值会永久减少，该值为0后该道具就没有用了。例如某道具的能量值为500，如果用它给一个战士补充了300的能量，则该道具的能量值变为200。现在你要带领你的战士们出征了，在出征前你要给这M个战士补充能量。假设初始时每个战士的能量都为0，补充完后所有的战士的能量都一样。如果一个战士在补充能量时只能使用1个道具（1个道具可以给若干个战士补充能量）。

现在请你写一段程序来计算一下，你最大能给每个战士补充的能量值是多少?

输入格式:

第一行为2个整数，分别代表N（1<=N<=10000）和M(1<=M<=20000)。

第二行为N个整数，代表这N个道具能提供的能量值（所有能量值大于等于100且小于等于2000000）。

输出格式:

为一个整数，代表你最大能给每个战士补充的能量值。测试数据保证有解。

输入样例:

```
5 13
765 506 483 329 492
```

输出样例:

```
164
```

代码长度限制	16 KB
时间限制	400 ms
内存限制	64 MB

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

bool check(int energy[], int n, int m, int energyPerWarrior) {
	int totalWarriors = 0;
	for (int i = 0; i < n; i++) {
		totalWarriors = energy[i] / energyPerWarrior+totalWarriors;
	}
	if(totalWarriors >= m) return true;
	else
		return false;
}

//right 一开始为 sum / m，没有能量浪费，最小为 0
```

```
//左闭右闭区间
int binarySearch(int energy[], int n, int m, int left, int right) {
    int mid;
     while (left <= right) {

            mid = left + (right - left) / 2; //  防止溢出
          if (check(energy, n, m, mid))//mid 可以，左边都可以
                left = mid + 1;
          else
                right= mid - 1;
     }
     return right;
}
/*左闭右开区间
int binarySearch(int energy[], int n, int m, int left, int right) {
    int mid;
     while (left < right) {

            mid = left + (right - left) / 2; //  防止溢出
          if (check(energy, n, m, mid))//mid 可以，左边都可以
                left = mid+1;
          else
                right= mid ;
     }
     return mid;
}

*/

int main() {
     int n, m;
     long long sum = 0;//超时危险
     scanf("%d%d", &n, &m);
     int* energy = (int*)malloc(sizeof(int) * n);


     for (int i = 0; i < n; i++) {
          scanf("%d", &energy[i]);
          sum += energy[i];
     }

     int maxEnergy = sum / m;
     printf("%d", binarySearch(energy, n, m, 1, maxEnergy));
     free(energy);
```

```
    return 0;
}
```