描述

乔治有一些同样长的小木棍，他把这些木棍随意砍成几段，直到每段的长都不超过50。

现在，他想把小木棍拼接成原来的样子，但是却忘记了自己开始时有多少根木棍和它们的长度。

给出每段小木棍的长度，编程帮他找出原始木棍的最小可能长度。

输入

输入文件共有二行。

第一行为一个单独的整数N表示砍过以后的小木棍的总数，其中N≤65

（管理员注：要把超过50的长度自觉过滤掉，坑了很多人了！）

第二行为N个用空个隔开的正整数，表示N根小木棍的长度。

输出

输出文件仅一行，表示要求的原始木棍的最小可能长度

分析

XJB剪枝系列QAQ

方法很简单，就是枚举木棍总长sum的大于_max的因子，dfs判断是否能满足题意，得到答案就退出枚举
认真分析一下，可以优化的地方有一下几点：

- 预处理出最长的木棍_max和最短的木棍_min
- 只枚举到sum / 2 ，如果仍未得到答案，直接输出sum
- dfs中得到答案，exit(0)直接退出程序 （不知道是否有优化）
- 如果有若干更小的木棍可以代替当前这根刚好凑成完整的木棍，那两者是可以互换的，则它们对能否构成完整木棍的贡献是相同的，不需要重复计算；并且，留下若干根短的后来可以有更加灵活的搭配，如果这种灵活搭配都不能满足条件而返回，那就没有继续算这个长度的意义了
- 当前组好的木棍长度为0对应的状态是尝试从头开始组成一根完整的木棍，如果

上面的递归调用能够正常返回到这里，就说明组成这个长度的木棍到最后不可行，如果可行程序直接就结束了，那只好返回了，没有继续计算的意义；而当当前组好的木棍长度不为0的时候，递归返回到这一层可能只是我们在这次循环中选择的木棍不合适，可能有其它合适的，就需要继续尝试了

这个程序有点贪心的意思，尽量先用比较长的，用短的灵活组合，如果这样都不能一直走到程序终点的话就说明这个长度不适合，所以这个程序没有打算走回头路，如果可行它就直接结束，不可行的时候不能直接结束递归，只能尝试逐层返回而不进行下次计算的方法让递归快速结束

代码

```
#include<bits/stdc++.h>
using namespace std;
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
const int INF = 0x3f3f3f3f;
typedef  long long LL;typedef  unsigned long long ULL;typedef  long double LDB;
inline LL CinLL(){LL x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}
inline int Cin(){int x=0,f=1;char ch=getchar();while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}while(isdigit(ch))x=x*10+ch-'0',ch=getchar();return f*x;}
int gun[55];
int n,x;
int _min = 51,_max = 1,sum;
int aim;
void dfs(int now,int be,int tot)
{
    if(tot == 0)
    {
        cout<<aim<<endl;
        exit(0);
    }
    _For(i,be,_min)
    {
        if(gun[i] && now+i <= aim)
```

```
        {
            gun[i]--;
            if(now + i == aim) {
                dfs(0,_max,tot-1);
                gun[i]++;
                return;
            }
            else dfs(now+i,i,tot);
            gun[i]++;
            if(now == 0) return;
        }
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin>>n;
    For(i,1,n)
    {
        cin>>x;
        if(x<=50) // 大于50的忽略掉
        {
            gun[x]++;
            sum+=x;
            _max = _max > x ? _max : x;
            _min = _min < x ? _min : x;
        }
    }
    int tmp = sum>>1;
    For(i,_max,tmp)
    {
        if(sum%i == 0)
        {
            aim = i;
            dfs(0,_max,sum/i);
        }
    }
    cout<<sum<<endl;
}

```