C - A->B

签到题，不难发现最优解就是一直-10到差值在$[0,9]$之间，然后特判0即可

```
int main() {
    int t;
    cin >> t;
    while (t--) {
        int a, b;
        cin >> a >> b;
        cout << (abs(a - b) + 9) / 10 << "\n";
    }
    return 0;
}
```

D - 删字符

贪心，从价值大到小不断删即可，最后根据索引还原字符串即可。

```
struct node{
    char x;
    int tmp;
    int z;
} a[N];

int cmp(node x,node y){
    return x.x > y.x;
}
int cmp1(node x,node y){
    return x.tmp < y.tmp;
}

void solve(){
    string ch,ss;
    cin >> ch;
    ss = ch;
    int n = ch.size();
    ch = "?" + ch;
    int w;
    cin >> w;
    int sum = 0;
    for(int i = 1;i <= n;i ++){
        sum += (ch[i] - 'a' + 1);
        a[i].x = ch[i];
        a[i].tmp = i;
        a[i].z = 0;
    }
    if(sum <= w){
        cout << ss << endl;
        return;
    }
```

```
    sort(a + 1,a + 1 + n,cmp);
    for(int i = 1;i <= n;i ++){
        sum -= (a[i].x - 'a' + 1);
        a[i].z = 1;
        if(sum <= w){
            break;
        }
    }
    sort(a + 1,a + 1 + n,cmp1);
    for(int i = 1;i <= n;i ++){
        if(a[i].z == 0) cout << a[i].x;
    }
    cout << endl;

}
```

E - 博弈

不难发现若字符串最后剩余两个不同的字符，那么一定是Alice获胜（一定可以拿走小的那个加入自己的字符串头部）。

深入考虑，对于存在$s[i] \neq s[i+1]$的情况，Alice只需不断保留这一对即可，而对于不存在$s[i] \neq s[i+1]$的情况，Alice也一定能拿到比Bob优势的情况。（此处证明不够严谨，也想不到什么严谨的证明）

所以Bob一定不可能赢，他要考虑的是能不能跟Alice平手。

由于字符串长度小于2000，考虑区间dp，若$dp[i][j]=1$表示区间为$[i,j]$的子串Alice为必胜态，反之为平手

所以若$s[i] \neq s[i+1]$，$dp[i][i+1]=1$，反之为0。

用区间dp的方式转移，从$[l,r]$转移，有如下可能

- Alice拿$s[l]$,Bob拿$s[l+1]$或$s[r]$
- Alice拿$s[r]$,Bob拿$s[r-1]$或$s[l]$

根据拿的情况进行dp转移即可，必胜态一定能转移到必胜态，若为平局态，考虑拿的两个的大小

- 若Alice无论怎么拿，Bob都能造成平局，即为平局态
- 若Alice拿其中一种，Bob都无法造成平局，即为必胜态。

时间复杂度为$o(n^2)$

```
void solve(){
    string s;
    cin >> s;
    int n = s.size();
    s = " " + s;
    vector<vector<int>> dp(n + 1,vector<int>(n + 1));
    for(int len = 2;len <= n;len += 2){
        for(int i = 1;i + len - 1 <= n;i ++){
            int j = i + len - 1;
            if(len == 2){
```

```
                if(s[i] != s[j]) dp[i][j] = 1;
                else dp[i][j] = 0;
            }else{
                if((dp[i + 1][j - 1] == 0 && s[j] <= s[i]) || (dp[i + 2][j] == 0
&& s[i + 1] <= s[i])){
                    if((dp[i + 1][j - 1] == 0 && s[i] <= s[j]) || (dp[i][j - 2]
== 0 && s[j - 1] <= s[j])){
                        dp[i][j] = 0;
                        continue;
                    }
                }
                dp[i][j] = 1;
            }
        }
    }
    if(dp[1][n] == 1){
        cout << "Alice" << endl;
    }else{
        cout << "Draw" << endl;
    }
}
```

继续深入思考不难发现，Bob平局的可能是无论$Alice$怎么拿，他都可以跟$Alice$拿到一样的字符，所以仅有两种可能平局。

- 形如$ABCCBA$这样的回文串，Bob只需拿根$Alice$对侧的字母即可。
- 形如$AABBCCDD$这样的字符串，Bob只需拿跟$Alice$同侧的字母即可。

时间复杂度为$o(n)$

```
void solve(){
    string s;
    cin >> s;
    int ok = 0;
    for(int i = 0,j = s.size() - 1;i < j;i ++,j --){
        if(s[i] != s[j]){
            for(int l = i;l < j;l += 2){
                if(s[l] != s[l + 1]){
                    ok = 1;
                    break;
                }
            }
            break;
        }
    }
    if(ok == 0){
        cout << "Draw" << endl;
        return;
    }
    cout << "Alice" << endl;
}
```

J - 这包是二分的啊

不难发现最后的结果一定是选择一个任务集合$S = a_1, a_2, .. a_i$第一天拿a_1，第二天拿a_2 ...第i天拿a_i，第$i+1$天拿a_1，如此循环，确定一个任务集合S后便可以通过条件判断这段任务集合是否能在给定天数内拿满金币，所以我们需要确定的是任务集合S的内容。

我们一定是从大到小拿，所以可以对数组降序排序后，直接二分集合$S = [a_1, a_{mid}]$，因为满足单调关系，所以能拿满向左二分，拿不满向右二分即可。

详情见代码

时间复杂度为$O(nlogn)$

```
void solve(){
    LL n,c,d;
    cin >> n >> c >> d;
    vector<LL> a(n + 1),s(n + 1);
    LL mx = 0;
    for(int i = 1;i <= n;i ++){
        cin >> a[i];
        mx = max(mx,a[i]);
    }
    sort(a.begin() + 1,a.end(),greater<int>());
    for(int i = 1;i <= n;i ++){
        s[i] = s[i - 1] + a[i];
    }
    if(s[min(d,n)] >= c){
        cout << "Infinity" << endl;
        return;
    }
    int l = 1,r = d,ans = -1;
    while(l <= r){
        LL mid = (l + r) / 2;
        LL res = s[min(n,mid)] * (d / mid) + s[min(n,d % mid)];
        if(res < c){
            r = r - 1;
        }else{
            ans = mid - 1;
            l = mid + 1;
        }
    }
    if(ans == -1){
        cout << "Impossible" << endl;
    }else{
        cout << ans << endl;
    }
}
```