1. Card game: 第一轮是给一堆牌，可以n个人一起玩。玩法就是开始时每个人有一堆，然后总是一起拿出最上面的那张，然后一起比大小。如果没有tie，那么最大的人把所有牌拿走。如果有tie，tie的人每人出三张牌作抵押，然后再每人拿出一张比大小，上面的步骤持续进行，知道只有一个赢家，把所有的牌都拿走。然后任何时候谁手上没牌了就输了，退出游戏。要求写一个函数，input是一堆list，每个list代表一个人手上的牌，返回谁会赢。抽牌的顺序是固定的，所以这是个没有随机因素的游戏。

Question: 如果不够怎么办？怎么处理拿回去的牌?

```
/*******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby, Perl,
Prolog, Javascript, Pascal, HTML, CSS, JS
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include<stdio.h>
#include<iostream>
#include<queue>
#include<vector>
using namespace std;

int getWinner(vector<queue<int>> hands) {
    vector<int> players;
    for(int i=0;i<hands.size();i++){
        if(!hands[i].empty()){
            players.push_back(i);
        }
    }

    while(players.size()>1){
        vector<int> cards;
        vector<int> winner;
        int wincard = -1;
        int cnt = players.size();
        for(int i:players){
            if(hands[i].front()>wincard){
                wincard = hands[i].front();
                winner.clear();
            }
            if(hands[i].front()==wincard){
                winner.push_back(i);
            }
            cards.push_back(hands[i].front());
```

```
            hands[i].pop();
        }

        while(winner.size()>1){
            for(int i:winner){
                for(int j=0;j<3&&!hands[i].empty();j++){
                    cards.push_back(hands[i].front());
                    hands[i].pop();
                }
            }
            wincard = -1;
            vector<int> tmp;
            for(int i:winner){
                if(hands[i].empty())
                    continue;
                if(hands[i].front()>wincard){
                    wincard = hands[i].front();
                    tmp.clear();
                }
                if(hands[i].front()==wincard){
                    tmp.push_back(i);
                }
                cards.push_back(hands[i].front());
                hands[i].pop();
            }
            winner = tmp;
        }
        for(int card:cards){
            hands[winner[0]].push(card);
        }

        vector<int> newplayers;
        for(int i:players){
            if(hands[i].empty()) continue;
            newplayers.push_back(i);
        }
        players = newplayers;
    }

    return players[0];
};

int helper(vector<vector<int>>& c){
```

```
    vector<queue<int>> Q;
    for(auto l:c){
        queue<int> tmp;
        for(auto card:l){
            tmp.push(card);
        }
        Q.push_back(tmp);
    }
    return getWinner(Q);
}

int main()
{
    vector<vector<int>> c1 = {{1,2,3,3,5},{1,2,3,4,4}};
    cout<<helper(c1)<<endl;
    return 0;
}
```

2. 设计题是tinyURL，不难但是followup很多，比方说界面长成啥样，如果request特别大怎么处理负载，以及怎么拿这个代码赚钱……好像之前看面经也有人被问到怎么赚钱了，第一轮是system design。问的是如何设计个app把一个很长很长的url变得很短。只能算是给了个比较basic的solution，跟面试官各种follow up讨论storage的问题。比较蛋疼的是面试官问我，有什么办法能用这个app赚钱？.....大哥了，我哪知道这破玩意怎么赚钱啊？？？更蛋疼的是问我，你有什么additional feature想加上去。.....一个把url变短的app还能有啥新的feature？？可能是我太没想象力了吧，反正随便说了几个。

http://systemdesigns.blogspot.com/search?q=tiny

Feistel cipher

/*********************************/

思路就是用a-zA-z0-9 这个62个字符 hash 原来的字符串。

```
#include<vector>
#include<string>
#include<unordered_map>
#include<cassert>
#include<iostream>
using namespace std;
class Solution {
public:
    string dict =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
    int id = 0;
    unordered_map<string,string> m;  //key is longURL, value is shortURL
```

```
    unordered_map<int,string> idm;  //key is id in DB, value is longURL
    // Encodes a URL to a shortened URL.
    string encode(string longUrl) {
        if(m.find(longUrl)!=m.end()){
            return m[longUrl];
        }
        int tmp = id;
        idm[id++] = longUrl;
        string ret = "";
        while(tmp>0){
            ret = dict[tmp%62] + ret;
            tmp /= 62;
        }
        while(ret.size()<6){
            ret = string(6 - ret.size(),'0') + ret;
        }
        return ret;
    }

    // Decodes a shortened URL to its original URL.
    string decode(string shortUrl) {
        int idx = 0;
        for(char c:shortUrl){
            idx = idx*62 + dict.find(c);
        }
        if(idm.find(idx)!=idm.end()){
            return idm[idx];
        }
        return "Unknown";
    }
};

int main(){
    vector<string> urls = { ...  };
    vector<string> shorts;
    Solution s;
    for(auto url:urls){
        string tmp = s.decode(s.encode(url));
        assert(tmp==url);
    }
    return 0;
}
```

3. 给了一 tree of resource, 每个resource 有parent, children resource.再给两 接口， grant(用户， resource), revoke(用户， resource) 如果 grant/revoke parent, 那么child resource也会被 g/r 要求实现 hasAccess(用户， resource).挂了

4. 一个是LZ77压缩算法，
https://gist.github.com/fogus/5401265

5. 一个是如何快速查找地图上某个disaster地点，其实就是实现一个quadtree。
Suppose query 是一个square区域?
https://www.geeksforgeeks.org/quad-tree/

6. 我的系统设计考的是设计Astroid Game，面试官说不要操心物理上的一些细节，主要是想考设计各个类，接口，总体game logic的思路。
https://gamedevelopment.tutsplus.com/tutorials/avoiding-the-blob-antipattern-a-pragmatic-approach-to-entity-composition--gamedev-1113
https://gamedevelopment.tutsplus.com/tutorials/create-a-simple-asteroids-game-using-component-based-entities--gamedev-1324

7. 第二轮，是在面试官Laptop上写的，写一个类似MongoDB pretty() method的函数。大概是下面这样，输入的JSON里有string, array, map三种类型。楼主当时用的是递归做，携带了一个level参数来决定缩进的量。

8. 第三轮是给几个sorted list，目标是返回在每个list里都出现的数字。开始假设每个list里没有重复，follow up是可以重复，而且对于重复的，output里要放在每个list里都出现的次数。举个例子。没有重复的情况，input可能是[1,2,3,4,5],[2,3,4,5],[4,7,8]那么output就是[4]，因为4在每个list里都出现了。再举个有重复的例子，input是[1,2,2,3,4,5,5,5],[2,3,3,4,5,5],[3,3,3,4,5,5]，那么output就是[3,4,5,5]

```
#include<iostream>
#include<vector>
#include <climits>
using namespace std;

vector<int> solve(vector<vector<int>>& lists){
    vector<int> ret;
    vector<int> idx(lists.size(),0);
    int val = INT_MIN;
    int cnt = 0;
    while(true){
    for(int i=0;i<lists.size();i++){
        if( idx[i]>=lists[i].size() )
            return ret;
```

```
        else{
            if(lists[i][idx[i]]>val) {
                val = lists[i][idx[i]];
                cnt = 1;
            }
            else if(lists[i][idx[i]]<val){
                while(idx[i]<lists[i].size()&&lists[i][idx[i]]<val){
                    idx[i]++;
                }
                if(idx[i]>=lists[i].size())
                    return ret;
                else if(lists[i][idx[i]]>val){
                    val = lists[i][idx[i]];
                    cnt = 1;
                }
                else{
                    cnt ++;
                    if(cnt==lists.size()){
                        ret.push_back(val);
                        cnt = 0;
                    }
                }
            }
            else{
                cnt ++;
                if(cnt==lists.size()){
                    ret.push_back(val);
                    cnt = 0;
                }
            }
        }
        idx[i]++;
    }
}

    return ret;
}

int main(){
    vector<vector<int>> lists = {{1,2,2,3,4,5,5,5},{2,3,3,4,5,5},{3,3,3,4,5,5}} ;
    vector<int> ret = solve(lists);
    for(auto i:ret){
        cout<<i<<",";
```

```
    }
    return 0;
}
```

9. 第二轮是coding，就是把一个非负整数变成英文，比如2变成two。除了edge case比较多以外没啥难的。

```
class Solution {
public:
    string numberToWords(int num) {
        int b = 1000000000;
        int m = 1000000;
        int t = 1000;
        if(num>=b){
            int rem = num%b;
            if(rem==0)
                return numberToWords(num/b) + " Billion";
            else
                return numberToWords(num/b) + " Billion " + numberToWords(rem);
        }
        else if(num>=m){
            int rem = num%m;
            if(rem==0)
                return numberToWords(num/m) + " Million";
            else
                return numberToWords(num/m) + " Million " + numberToWords(rem);
        }
        else if(num>=t){
            int rem = num%t;
            if(rem==0)
                return numberToWords(num/t) + " Thousand";
            else
                return numberToWords(num/t) + " Thousand " + numberToWords(rem);
        }
        else if(num>=100){
            int rem = num%100;
            if(rem==0)
                return under20[num/100] + " Hundred";
            else
                return under20[num/100] + " Hundred " + numberToWords(rem);
        }
        else if(num>=20){
            int rem = num%10;
            if(rem==0)
```

```
            return tens[num/10];
        else
            return tens[num/10] + " "+ numberToWords(rem);
    }
    else {
        return under20[num];
    }
  }
private:
  vector<string> under20 = {
    "Zero","One","Two","Three","Four","Five","Six","Seven","Eight","Nine",
    "Ten","Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen",
    "Eighteen", "Nineteen"};
  vector<string> tens = {
    "", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"
  };
};
```

10. The Skyline Problem，leetcode的题，之前没见过，直接挂了

Divide and conquer O(nlogn)

```
class Solution {
        public:
  vector<pair<int,int>> getSkyline(vector<vector<int>>& buildings) {
                vector<pair<int,int>> ret;
    if (buildings.empty())
                        return ret;
                return recurSkyline(buildings, 0, buildings.size() - 1);
        }

        private:
  vector<pair<int,int>> recurSkyline(vector<vector<int>>& buildings, int p, int q) {
                if (p < q) {
                        int mid = p + (q - p) / 2;
                        return merge(recurSkyline(buildings, p, mid),
                                        recurSkyline(buildings, mid + 1, q));
                } else {
                        vector<pair<int,int>> rs =
        {pair<int,int>(buildings[p][0], buildings[p][2]),pair<int,int>(buildings[p][1], 0 )};
                        return rs;
                }
        }
```

```
        vector<pair<int,int>> merge(vector<pair<int,int>> l1, vector<pair<int,int>> l2) {
                vector<pair<int,int>> rs;
                int h1 = 0, h2 = 0;
                int i=0,j=0;
        while (i<l1.size()  && j<l2.size()) {
                        int x = 0, h = 0;
                        if (l1[i].first < l2[j].first) {
                                x = l1[i].first;
                                h1 = l1[i].second;
                                h = max(h1, h2);
                                i++;
                        } else if (l1[i].first > l2[j].first) {
                                x = l2[j].first;
                                h2 = l2[j].second;
                                h = max(h1, h2);
                                j++;
                        } else {
                                x = l1[i].first;
                                h1 = l1[i].second;
                                h2 = l2[j].second;
                                h = max(h1, h2);
                                i++;
                                j++;
                        }
                        if (rs.empty() || h != rs.back().second) {
                                rs.push_back(pair<int,int>(x,h));
                        }
                }
                while(i<l1.size())
        rs.push_back(l1[i++]);
                while(j<l2.size())
        rs.push_back(l2[j++]);
                return rs;
        }
};
```

O(nlogn) sort

```
class Solution {
public:
    vector<pair<int, int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> h, res;
        multiset<int> m;
        int pre = 0, cur = 0;
        for (auto &a : buildings) {
```

```
            h.push_back({a[0], -a[2]});
            h.push_back({a[1], a[2]});
        }
        sort(h.begin(), h.end());
        m.insert(0);
        for (auto &a : h) {
            if (a.second < 0) m.insert(-a.second);
            else m.erase(m.find(a.second));
            cur = *m.rbegin();
            if (cur != pre) {
                res.push_back({a.first, cur});
                pre = cur;
            }
        }
        return res;
    }
};

class Solution {
public:
    vector<pair<int, int>> getSkyline(vector<vector<int>>& buildings) {
        sort(buildings.begin(),buildings.end());
        int cur_X = 0, cur_H = 0, cur = 0, len = buildings.size();
        vector<pair<int,int>> res;
        priority_queue<pair<int,int>> Q;
        while(cur<len || !Q.empty()){
            if(Q.empty()|| cur<len && buildings[cur][0] <= Q.top().second){
                cur_X = buildings[cur][0];
                while(cur<len && buildings[cur][0]== cur_X){
                    Q.push(pair<int,int>(buildings[cur][2],buildings[cur][1]));
                    cur ++;
                }
            }
            else {
                cur_X = Q.top().second;
                while(!Q.empty()&&Q.top().second<= cur_X)
                    Q.pop();
            }
            cur_H = Q.empty() ? 0:Q.top().first;
            if( res.empty()||(res.back().second != cur_H))
                res.push_back(pair<int,int>(cur_X,cur_H));
        }
        return res;
    }
};
```

11. 去了之后先介绍公司，小哥语气平缓有点像念经，我面试的职位好像是software engineering focus on full stack / front end dev （因为我hr面时吹嘘了太多我的前端技能）所以和别人的面试流程不太一样，一般genereal sde是三轮白板，我是一轮白板加一轮上机编程，js写web application。。白板design国际象棋，上机大概就是http request读一些数据，然后visualize展示出来在网页上，我用的reactjs，结果开始写了发现react都忘光了。。一边stackoverflow一边查doc写，场面一度十分尴尬。

12. 白板design国际象棋