玩牌高手

题目描述：

给定一个长度为 n 的整型数组，表示一个选手在 n 轮内可选择的牌面分数。选手基于规则选牌，请计算所有轮结束后其可以获得的最高总分数。选择规则如下：

1、在每轮里选手可以选择获取该轮牌面，则其总分数加上该轮牌面分数，为其新的总分数。

2、选手也可不选择本轮牌面直接跳到下一轮，此时将当前总分数还原为 3 轮前的总分数，若当前轮次小于等于 3（即在第 1、2、3 轮选择跳过轮次），则总分数置为 0。

3、选手的初始总分数为 0，且必须依次参加每一轮。

输入描述：

第一行为一个小写逗号分割的字符串，表示 n 轮的牌面分数，1<= n <=20。

分数值为整数，-100 <= 分数值 <= 100。

不考虑格式问题。

输出描述：

所有轮结束后选手获得的最高总分数。

补充说明：

示例 1

输入：

```
1,-5,-6,4,3,6,-2
```

输出：

```
11
```

说明：

总共有 7 轮牌面。
第一轮选择该轮牌面，总分数为 1。
第二轮不选择该轮牌面，总分数还原为 0。
第三轮不选择该轮牌面，总分数还原为 0。
第四轮选择该轮牌面，总分数为 4。
第五轮选择该轮牌面，总分数为 7。
第六轮选择该轮牌面，总分数为 13。
第七轮如果不选择该轮牌面，则总分数还原到 3 轮 1 前分数，即第四轮的总分数 4，如果选择该轮牌面，总分数为 11，所以选择该轮牌面。
因此，最终的最高总分为 11。

```
#include <iostream>
#include <vector>
using namespace std;
int getres(vector<int> &nums)
{
    int sum=0;
    int size=nums.size();
```

```
    vector<int> dp(size);
    dp[0]=(0>nums[0])?0:nums[0];
    for(int i=1;i<size;++i)
    {
        if(i<3)
        {
            int tmpnum=dp[i-1]+nums[i];
            dp[i]=tmpnum < 0 ? 0 :tmpnum;
        }else{
            dp[i]=max(dp[i-1]+nums[i],dp[i-3]);
        }
    }
    return dp[size-1];
}
int main()
{
    string str;
    cin>>str;
    int len=str.size();
    //dp[i]=max(0,dp[i-1]+nums[i]);
    //dp[i]=max(dp[i-1]+nums[i],dp[i-3]); i>=3
    int index=str.find(',');
    int lastindex=-1;
    vector<int> nums;
    while(index!=-1)
    {
        string tmp=str.substr(lastindex+1,index-lastindex-1);
        nums.push_back(atoi(tmp.c_str()));
        lastindex=index;
        index = str.find( ',',lastindex + 1);
    }
    string tmp=str.substr(lastindex+1);
    nums.push_back(atoi(tmp.c_str()));
    cout<<getres(nums)<<endl;
    return 0;
}
```