**Programming**

10. “default” means we want to use compiler-generated version of function, so don’t need to specify a body.

Here are cases we need default constructor:

When we want to force compiler to generate constructor.

When all parameters have default values;

11.<http://silviuardelean.ro/2012/06/05/few-singleton-approaches/>

class safeSingleton {

static std::shared\_ptr< safeSingleton > instance\_;

static std::once\_flag only\_one;

safeSingleton(int id) {

std::cout << "safeSingleton::Singleton()" << id << std::endl;

}

safeSingleton(const safeSingleton& rs) {

instance\_ = rs.instance\_;

}

safeSingleton& operator = (const safeSingleton& rs) {

if (this != &rs) {

instance\_ = rs.instance\_;

}

return \*this;

}

public:

~safeSingleton() {

std::cout << "Singleton::~Singleton" << std::endl;

}

static safeSingleton & getInstance( int id ) {

std::call\_once( safeSingleton::only\_one, [] (int idx) {

safeSingleton::instance\_.reset( new safeSingleton(idx) );

std::cout << "safeSingleton::create\_singleton\_() | thread id " + idx << std::endl;

}, id );

return \*safeSingleton::instance\_;

}

void demo(int id) {

std::cout << "demo stuff from thread id " << id << std::endl;

}

};

std::once\_flag safeSingleton::only\_one;

std::shared\_ptr< safeSingleton > safeSingleton::instance\_ = nullptr;

12.<https://leetcode.com/discuss/15153/a-clean-dp-solution-which-generalizes-to-k-transactions>

class Solution {

public:

int maxProfit(vector<int> &prices) {

// f[k, ii] represents the max profit up until prices[ii] (Note: NOT ending with prices[ii]) using at most k transactions.

// f[k, ii] = max(f[k, ii-1], prices[ii] - prices[jj] + f[k-1, jj]) { jj in range of [0, ii-1] }

// = max(f[k, ii-1], prices[ii] + max(f[k-1, jj] - prices[jj]))

// f[0, ii] = 0; 0 times transation makes 0 profit

// f[k, 0] = 0; if there is only one price data point you can't make any money no matter how many times you can trade

if (prices.size() <= 1) return 0;

else {

int K = 2; // number of max transation allowed

int maxProf = 0;

vector<vector<int>> f(K+1, vector<int>(prices.size(), 0));

for (int kk = 1; kk <= K; kk++) {

int tmpMax = f[kk-1][0] - prices[0];

for (int ii = 1; ii < prices.size(); ii++) {

f[kk][ii] = max(f[kk][ii-1], prices[ii] + tmpMax);

tmpMax = max(tmpMax, f[kk-1][ii] - prices[ii]);

maxProf = max(f[kk][ii], maxProf);

}

}

return maxProf;

}

}

};

13:<http://javabypatel.blogspot.com/2015/09/find-element-that-appears-once.html>

Key point is: n^0=n

public class FindElementThatAppearOnceRestAllAppearTwice {

public static void main(String[] args) {

int arr[] = {6, 2, 5, 10, 5, 2, 10};

System.out.println(findElementThatAppearOnce(arr));

}

private static int findElementThatAppearOnce(int arr[]){

int result = 0;

for (int i=0; i<arr.length; i++){

result = result ^ arr[i];

}

return result;

}

}

14:<https://isocpp.org/wiki/faq/strange-inheritance#calling-virtuals-from-base>

Yes. It’s sometimes (not always!) a great idea. For example, suppose all Shape objects have a common algorithm for printing, but this algorithm depends on their area and they all have a potentially different way to compute their area. In this case Shape’s area() method would necessarily have to be virtual (probably pure virtual) but Shape::print() could, if we were guaranteed no derived class wanted a different algorithm for printing, be a non-virtual defined in the base class Shape.

#include "Shape.h"

void Shape::print() const

{

float a = this->area(); // area() is pure virtual

// ...

}

15:<https://en.wikipedia.org/wiki/Random_number_generation>

uses the recurrence

to generate numbers, where *a*, *b* and *m* are large integers, and is the next in *X* as a series of pseudo-random numbers. The maximum number of numbers the formula can produce is the [modulus](https://en.wikipedia.org/wiki/Modulus_%28algebraic_number_theory%29), *m*.

16.<http://cs.stackexchange.com/questions/29204/how-to-simulate-a-die-given-a-fair-coin>

2k=6m

Coin toss k times, each time represent a bit of an Integer, then use the result integer to mod 6

17. 题目意思没表达清楚,给人感觉是永远执行相同指令。<https://en.wikibooks.org/wiki/Puzzles/Logic_puzzles/Parachuted_Robots>

Google了下，原题意思是同时“开始”执行程序

void roboMeet() {

bool reachedZero = false;

while( !meet() ) {

if(reachedZero) {

moveLeft();

moveLeft();

} else {

moveLeft();

moveLeft();

moveRight();

}

if( at-zero() ) {

reachedZero = true;

}

}

}