Programming Design, Spring 2015

Suggested Solution for Homework 8

Solution provider: Tammy Chang

Problem 1

(a)

```
void LotteryMachine::initialize(){
    tokenCount = 0;
    outcomeCount = 0;
    tokens = NULL;
    outcomes = NULL;
}
void LotteryMachine::release(){
    delete[] tokens;
    delete[] outcomes;
    tokens = NULL;
    outcomes = NULL;
}
```

(b)

```
void LotteryMachine::initialize(int n,int t[]){
    tokenCount= n;
    outcomeCount = 0;
    outcomes = NULL;
    tokens = new int[n];
    for(int i=0; i<n; i++){
       tokens[i] = t[i];
    }
}
```

(c)

```
bool LotteryMachine::draw(int m){

    if(m<0 || m>tokenCount){
       return false;
```

```
    }
    else{
        // copy tokens and tokenCount
        int* copyTokens = new int[tokenCount];
        for(int c=0; c<tokenCount; c++){
            copyTokens[c] = tokens[c];
        }
        int copyTokenCount = tokenCount;

        srand(time(0));
        int r = 0;
        delete[] outcomes;
        outcomes = new int[m];
        // when the tokens drawn aren't enough
        while(outcomeCount<m){
            r = rand()%copyTokenCount;
            // copy the token to outcomes
            outcomes[outcomeCount] = copyTokens[r];
            // replace the drawn token value to the last token
            copyTokens[r] = copyTokens[copyTokenCount-1];
            // the range of random number decrease one
            copyTokenCount--;
            outcomeCount++;
        }
        delete[] copyTokens;
        return true;
    }
}
```

(d)

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
```

```
class LotteryMachine{
public:
    LotteryMachine();
    LotteryMachine(int n, int t[]);
    bool draw(int m);
    ~LotteryMachine();
private:
    int tokenCount;
    int* tokens;
    int outcomeCount;
    int* outcomes;

};
// constructor
LotteryMachine::LotteryMachine(){
    tokenCount = 0;
    outcomeCount = 0;
    tokens = NULL;
    outcomes = NULL;
}
LotteryMachine::LotteryMachine(int n,int t[]){
    tokenCount= n;
    outcomeCount = 0;
    outcomes = NULL;
    tokens = new int[n];
    for(int i=0; i<n; i++){
        tokens[i] = t[i];
    }
}
// destructor
LotteryMachine::~LotteryMachine(){
    delete[] tokens;
    delete[] outcomes;
    tokens = NULL;
    outcomes = NULL;
```

```
}
bool LotteryMachine::draw(int m){
    if(m<0 || m>tokenCount){
        return false;
    }
    else{
        // copy tokens and tokenCount
        int* copyTokens = new int[tokenCount];
        for(int c=0; c<tokenCount; c++){
            copyTokens[c] = tokens[c];
        }
        int copyTokenCount = tokenCount;

        srand(time(0));
        int r = 0;
        delete[] outcomes;
        outcomes = new int[m];
        // when the tokens drawn aren't enough
        while(outcomeCount<m){
            r = rand()%copyTokenCount;
            // copy the token to outcomes
            outcomes[outcomeCount] = copyTokens[r];
            // replace the drawn token value to the last token
            copyTokens[r] = copyTokens[copyTokenCount-1];
            // the range of random number decrease one
            copyTokenCount--;
            outcomeCount++;
        }
        delete[] copyTokens;
        return true;
    }
}
```

Problem 2

Please see the 150901.cpp file.

Problem 3

Please see the 150902.cpp file.