PIC 10A 1C. Week 7b Problems. TA: Eric Kim [Solutions]

1. I Know Kung-Fu...

Write a program that does the following:

(1) Asks the user for the number of desired rows and columns of the matrix.

(2) Generate a matrix of the desired size. Each entry is a random integer ranging from 1 to 6, inclusive.

(3) Displays each entry of the matrix in the following style:

```
1 4 2 5
3 4 6 1
5 4 2 3
```

Example Output:

```
Number of rows? 3
Number of cols? 4
1 4 2 5
3 4 6 1
5 4 2 3
```

```
// YOUR CODE HERE. Use the vector class. Try using a nested for-loop too.
```

[Solution]

```
#include <iostream>
#include <vector>
#include <cstdlib> // rand, srand
#include <time>
using namespace std;
int main() {
    int n, m;
    srand(time(nullptr));
    cout << "Number of rows? ";    cin >> n;
    cout << "Number of cols? ";    cin >> m;
    vector< vector<int> > M(n, vector<int>(m, 0));
    /* Initialize matrix values: [1,6] */
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            M[i][j] = (rand() % 6)+1;
        }
    }
    /* Make matrix look pretty */
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            cout << M[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

2. Evenly odd, or oddly even?

Write a program that generates 500 random integers, and outputs the percentage that are even. Format the output so that the percentage is in fixed-point notation, with at most 2 decimal points. This is a quick test to see if there is some bias in rand()'s PRNG algorithm. For reproducibility, set the PRNG seed to: 42.

Expected Output:

```
    Percentage of even: 45.60%
```

```
// YOUR CODE HERE
```

[Solution]

```
#include <iostream>
#include <cstdlib> // rand, srand
#include <iomanip>
using namespace std;
int main() {
    unsigned int nbevens = 0;
    const int N = 500;
    srand(42);
    for (int i = 0; i < N; ++i) {
        int r = rand();
        if ((r % 2) == 0)
            nbevens += 1;
    }
    double frac = static_cast<double>(nbevens) / N;
    cout << "Percentage of even: ";
    cout << fixed << setprecision(2) << 100.0*frac << "%" << endl;
    return 0;
}
```

3. Magnitude: Pop-pop!

After each code snippet, what are the contents of the vector? If a crash occurs, explain why. The first has been done for you.

Code	**Vector Contents**
`vector<int> v0 = {1, 2};` `v0.push_back(3);`	`[1, 2, 3]`
`vector<int> v1(4);` `v1.pop_back();`	[0, 0, 0] int default-initializes to 0.
`vector<char> v2(3, 'a');`	['a', 'a', 'a']
`vector<int> v3 = {1, 2};` `v3.pop_back(); v3.pop_back();`	[] (Empty)
`vector<double> v4 = {};` `v4.pop_back();`	Runtime Error! Called pop_back() on an empty vector.
`vector<string> v5(2);`	["", "meow"]

`v5[1] = "meow";`	string default-initializes to empty string "".

4. Iterators

Consider the following for-loop that squares all of the even numbers within a vector:

```
vector<int> v = {1, 2, 3, 4, 5, 6};
for( size_t i = 0; i < v.size(); ++i ) {
    if ((v[i] % 2) == 0) {
        v[i] = v[i]*v[i];
    }
}
```

Rewrite the above loop but using iterators (ie using begin, end, vector<int>::iterator, etc.).

```
// YOUR CODE HERE
```

[Solution]

```
for (vector<int> v::iterator iter = begin(v); iter!=end(v); ++iter) {
    if ((*iter)%2) == 0)
        (*iter) = (*iter)*(*iter);
}
```

5. Timey-Wimey

Recall the following documentation for the `time()` function (defined in `<ctime>`):

```
/**
 * This function takes in a pointer to a time_t variable, and assigns to that
 * variable the number of seconds since Jan 1, 1970 00:00 UTC, then returns
 * this value as a time_t variable. Given a nullptr input, it simply returns
 * the number of seconds since Jan 1st, 1970.
 * @param timeptr the pointer to time_t
 * @return the number of seconds elapsed since Jan. 1st, 1970.
 **/
time_t time(time_t *timeptr);
```

Louis Reasoner wants to write a program that times the number of seconds between user inputs. For instance, the program should behave as follows:

```
    Press enter to start the timer!
    <you pressed enter!>
    Press enter again to stop the timer...
    <you pressed enter!>
    Time elapsed: 12 seconds.
```

Does the following program behave correctly? If not, explain why.

```
#include <ctime>
#include <iostream>
using namespace std;
int main() {
  time_t t;
  cout << "ENTER to start the timer!" << endl;
  cin.get(); // Wait for user to hit enter
  cout << "<you pressed enter!>" << endl;
  time(&t); // Initialize timer t
  cout << "Press enter again to stop the timer...\n";
```

```
  cin.get(); // Wait for user to hit enter again
  cout << "<you pressed enter!>" << endl;
  time_t duration = time(nullptr) - t;
  cout << "Time elapsed: " << duration << " seconds.\n";
  return 0;
}
```

[Solution]

It's correct! Good job Louis.

6. There Is No Compiler...

For the following, write down the expected output. If an error occurs, describe why. Assume all headers have been included, and that we are using the standard namespace:

`int d = 42;`
`int* dp = &d;`
`*dp = 43;`
`cout << *dp << " " << d;`
	43 43

`double x = 42.4;`
`(*x) = (*x) + 1;`
`cout << x;`
	Error: x is not a pointer, can't dereference!

`double x = 13.42;`
`double &xr = x;`
`xr = xr + 1;`
`cout << xr << " " << x;`
	14.42 14.42

`double x = 1.1;  double y = 3.2;`
`double &xr = x;`
`xr = &y;`
`xr += 1;`
`cout << xr << " " << y;`
	Error: Can't change what a reference points to! Ie "xr = &y;" is an error.

`int x = 1;  int y = 3;`
`int *xr = &x;`
`xr = &y;`
`xr += 1;`
`cout << xr << " " << y;`
	002DFDC4 3
	Note: When we do "xr = &y", we assign the address of y into xr. Thus, "002DFDC4" is displayed (or whatever the address of y happens to be).

`int x = 1;  int y = 3;`
`int *xr = &x;`
`xr = &y;`
`(*xr) += 1;`
`cout << *xr << " " << y;`
	4 4

`string s("Spoon");`
`string *sp = &s;`
`cout << sp.size();`
	Error: sp is a pointer, not an object! Can't use dot-notation on pointer. Should have done:
	`cout << sp->size();`
	Recall: sp->size() is equivalent to: (*sp).size()

7. Best Function Names NA

Consider the following function definitions/declarations. If there are any issues, explain why. Assume all relevant headers are included, and we are using the standard namespace.

`double foo(int a, string s) {`
`    return s[a + 1];`
`}`
Error: foo should return a double, but returns a char.
	`void bar(int x) {`
	`    return x + 1;`
	`}`
	Error: bar is declared as returning void, but here it returns an int.

`char baz(string s) {`
`    return s[0];`
`}`
Correct! s[0] is indeed a char.
	`string garply(string str, int n) {`
	`    return s[n];`
	`}`
	Error: s[n] is a char, but garply should return a string!

`void zzz() {`
`}`
This is fine, no issue.
	`string hmm() {`
	`    return 'a';`
	`}`
	Error: 'a' is a char, but hmm() should return a string!

`double meow(int x) {`
`    return x;`
`}`
This is OK - C++ will simply cast the "int x" into a double.
	`int bark(double d) {`
	`    return d;`
	`}`
	This is OK - however, C++ will cast the "double d" into an integer, ie truncate, so you may lose information, ie 3.14 -> 3.

8. What's your Vector, Victor?

(a)Write some code that displays the contents of a vector containing integers called "vec":

```
    vector<int> vec = {1, 2, 4, 5};
```

Your code should display:

```
    [1, 2, 4, 5]
```

[Solution]

```
cout << "[";
for (int c : vec) {
    cout << c << ", ";
}
/* Remove the trailing ", ", then add final "]" */
cout << "\b\b]";
```

(b) Write code that removes the odd integers from vec, ie if vec is [1, 2, 3, 5, 8], then after running your code, vec should be [2, 8].

[Solution]

```
vector<int>::iterator iter = begin(vec);
while (iter != end(vec)) {
    if ((*iter) % 2 == 1) {
        vec.erase(iter);
    } else {
        ++iter;
    }
}
```

(c) Consider the following implementation of part b:

```
size_t i = 0;
while (i < vec.size()) {
    if ((vec[i] % 2) == 1) {
        vec.erase(begin(vec) + i);
    }
    i += 1;
}
```

Does this code run correctly? If not, what is vec after this code is run? (Assume vec starts as: [1,2,3,5,8]).

[Solution]

No, this code is incorrect. It will skip all numbers that are immediately after an odd integer - thus, vec would be [2,5,8], rather than [2,8].