CS 7A **Name (Print):** _____

Quiz 3 Solutions

Write all responses on separate paper. Do not use a computer.

1. (50 points) Find and remove all errors in each program. When you have removed those bugs, the resulting program will compile, run, and write "Success!" Even if you think you have spotted an error, you still need to enter the (original, unimproved) program fragment and test it; you may have guessed wrong about what the error is, or there may be more errors in a fragment than you spotted.

   (a) `int x = 2000; char c = x; if (c==2000) cout << "Success!\n";`

   This is a narrowing error caused by the integer value larger than 255 being assigned to to a variable of type `char`. The best thing to do is avoid this sort of error, but you could also cast 2000 as a `char` in the comparison like so:

```
int x = 2000;
    char c = x; /// narrowing error
    if (c==char(2000))
        cout << "Success!\n";
```

   (b) `vector v(5); for (int i=0; i<=v.size(); ++i) ; cout << "Success!\n";`

   It's not obvious why the vector `v` is introduced here. The body of the for loop is empty, so i will count to 5 without anything else being done, and then we print `"Success!"` to the console. The simplest fix for this would be to simply cout ≪ "Success!\n", if you did want to use a vector, first it would be a vector of `char` and you'd want to initialize it to have the characters in the string `"Success!"`.

   (c) `int i=0; int j = 9; while (i<10) ++j; if (j<i) cout << "Success!\n";`

   This contains an infinite loop, since `i` is initialized to 0 and the body of the while loop (`++j`) doesn't change `i`. If we simply change the body of the loop to increment `i` instead of `j`, then, while it is kind of silly, the code will achieve the desired result.

   (d) `string s = "Success!\n"; for (int i=0; i<10; ++i) cout << s[i];`

   This doesn't have any errors.

   (e) `int x = 2; double d = 5/(x2); if (d==2*x+0.5) cout << "Success!\n";`

   That minus sign was missing between `x` and `2` here, so I'll put it in and answer that more sensible question. This then leads to a divide by zero error, but we can fix that by initializing `x` to something else, say `x = 4`. Now we can change the value of `d` and the comparison so that the condition is true and we get `Success!`. If `x=4` is an `int` then `2*x+0.5` promotes it to a double (or a float?) and so `2*4+0.5` or 8.5 is assigned to `d`. `x`, of course, is still and `int`, so the expression `17/(x-2)` would be integer division and be truncated to 8, so for the comparison with `d` to be true, force a promotion to double by involving a decimal point like so:

```
int x = 4;
double d = 17./(x-2);
if (d==2*x+0.5) cout << "Success!\n";
```

   (f) `string<char> s = "Success!\n"; for (int i=0; i<=10; ++i) cout << s[i];`

   `string<char>` is not any kind of proper syntax in C++. You could go with a `vector` of `char` like so:

```
vector<char> s{'S','u','c','c','e','s','s','!','\n'};
for (int i=0; i<=10; ++i)
    cout << s[i];
```

   ...or just a `string`, like this:

```
string s = "Success!\n";
for (int i=0; i<=10; ++i)
    cout << s[i];
```

   (g) `int i=0; while (i<10) ++j; if (j<i) cout << "Success!\n";`

   Here there are two problems: (1) `j` is not declared or initialized and (2) the while loop has no way to terminate. One way to fix this:

```
int i=0, j{0};
while (i<10) ++i;
if (j<i) cout << "Success!\n";
```

(h) `int x = 4; double d = 5/(x2); if (d=2*x+0.5) cout << "Success!\n";`
That minus sign was missing between `x` and `2` here, so I'll put it in and answer that more sensible question. This then successfully prints out "`Success`", but obviously the assignment statement in the conditional should be a comparison. Making it a comparison means it no longer prints "`Success`" because `d` is 2, not 8.5, so you could change the code like so to get the comparison to be true:

```
int x = 4;
double d = 5/(x-2);
if (d==2) cout << "Success!\n";
```

2. (50 points) Write a function that takes an `int N` and returns a `vector<int>` which contains all the square-free numbers less than `N`, so that if, for instance, `N = 19` then the function returns the vector {2,3,5,6,7,10,11,13,14,15,17}

```
// GH square free generator

#include "..\std_lib_facilities.h"

bool isSquareFree(int);
void buildSquareFree(vector<int>&,int);
void show(vector<int>);

int main() {
    int N{0};
    vector<int> sfree;
    cout << "\nEnter␣N␣to␣see␣all␣the␣square-free␣numbers␣less␣than␣N:␣";
    while(cin>>N) {
        buildSquareFree(sfree,N);
        show(sfree);
        cout << "\nEnter␣N␣to␣see␣all␣the␣square-free␣numbers␣less␣than␣N:␣";
    }
}

void buildSquareFree(vector<int>& sFree, int N) {
    sFree.clear();
    for(int i = 2; i < N; ++i)
        if(isSquareFree(i)) sFree.push_back(i);
}

void show(vector<int> sF) {
    for(int i = 0; i < sF.size(); ++i) {
        cout << sF[i] << "␣";
        if((i+1)%14==0) cout << endl;
    }
}

bool isSquareFree(int N) {
    for(int i = 2;i<=sqrt(N);++i)
        if(N%(i*i)==0) // evenly divisible by a square
            return false;
    return true;
}
```