

SI@UCF Computer Science Camp
Intro to Competitive Programming in C++
Quiz #2 Solutions
Date: 6/17/2026

1) (15 pts) Let a map store frequency information as follows: the key to the map is the item being stored and the matching value for a key is the number of times that item is stored. We can describe an unordered list of integers in this manner. For example the map $\{(3, 5), (7, 2), (8, 1)\}$ is storing the list of integers 3, 3, 3, 3, 3, 7, 7, 8. Consider the problem of being given two unordered lists of integers stored as maps in this manner and being asked how many ways we can choose one value from the first list and one value from the second list and add up to a target value, target, that we are given. Complete the function below so that it runs in $O(nm)$ time, where n is the size of map1, and m is the size of map2, and returns the number of pairs of integers (one from the first list and one from the second list) that add up to target. As an example, if the input were the first map shown above and the map $\{(1, 7), (5, 4), (8, 3)\}$ and our target was 8, then the answer would be $5 \times 4 + 2 \times 7 = 34$. (We can pick any of the 5 copies of 3 from the first list and pair it with any of 4 copies of 5 from the second list.) **Note: This solution is meant to be slow and have a double loop structure with the run-time designated before. The improvement to this solution will be the next question.**

```
using namespace std;
#include <bits/stdc++.h>
typedef long long ll;
```

```
ll nummatchesslow(map<int,int> map1, map<int,int> map2, int target) {
```

```
    ll res = 0; // 1 pt
    for (auto x=map1.begin(); x!=map1.end(); x++) // 3 pts
        for (auto y=map2.begin(); y!=map2.end(); y++) // 3 pts
            if ( ((*x).first + (*y).first) == target ) // 3 pts
                res += ( ((*x).second) * ((*y).second)); // 4 pts

    return res; // 1 pt
```

```
}
```

2) (15 pts) Alter the solution in question #1 so that the run time is now either $O(n \lg m)$ or $O(m \lg n)$. (Hint: The run time of finding a particular key in a map of size n is $O(\lg n)$.)

```
using namespace std;
#include <bits/stdc++.h>
typedef long long ll;

ll nummatchesfast(map<int,int> map1, map<int,int> map2, int target) {

    ll res = 0; // 1 pt
    for (auto x=map1.begin(); x!=map1.end(); x++) { // 3 pts
        int need = target - (*x).first; // 2 pts
        auto tmp = map2.find(need); // 2 pts
        if (tmp != map2.end()) // 2 pts
            res += ((*x).second)*((*tmp).second); // 4 pts
    }

    return res; // 1 pt
}
```

3) (5 pts) What is the output of running the function call $f(4)$ where f is the function defined below?

```
void f(int n) {
    if (n == 0) return;
    f(n-1);
    cout << n;
    f(n-1);
}
```

121312141213121

Grading: 5 pts if correct

3 pts if the frequency of 1, 2, 3, 4 are all correct but in the wrong order

2 pts if each integer from 1 to 4 appears at least once

0 otherwise

4) (15 pts) Complete the function below so that it takes in a positive integer n , and returns a vector of pairs of integer, such that each pair multiplies to exactly n , with the first value in the pair less than or equal to the second value of the pair. You should return a vector that contains all distinct pairs that satisfy the criteria. The run-time of your code should be $O(\sqrt{n})$.

```
vector<pair<int,int>> factor(int n) {  
  
    vector<pair<int,int>> res;                // 2 pts  
  
    for (int i=1; i*i<=n; i++)                // 4 pts  
        if (n%i == 0)                         // 3 pts  
            res.push_back(pair<int,int>{i,n/i}); // 5 pts  
    return res;                               // 1 pt  
  
}
```

5) (10 pts) You were shown an algorithm that goes through each permutation of the list 0, 1, 2, ..., $n - 1$, for a given number n . This algorithm goes through the permutations in a particular order. What is the permutation the algorithm generates right after it generates the permutation

3, 1, 6, 0, 9, 5, 8, 7, 4, 2

3, 1, 6, 0, 9, 5, 8, 7, 4, 2

Grading: 1 pt per slot

6) (25 pts) Consider the following problem:

You are given a list of word lengths, in the order that the words must be printed. You know that you must print all of the text in at most n lines, and that there must be a space inserted between all adjacent words on the same line. For example, if the word lengths were 10, 3, 8, 5, 6, 7, 2, we could fit the text on two lines with width 23 by putting (10, 3, 8) on the first line and (5, 6, 7, 2) on the second line. Notice that $10 + 1 + 3 + 1 + 8 = 23$ and $5 + 1 + 6 + 1 + 7 + 2 = 23$ as well. (Thus, no space is necessary between the word of length 8 and the word of length 5 since they are on two different lines.) Given this information, determine the minimum width of text necessary to completely fit the text in at most n lines. To solve the problem, write a function that takes in a vector<int> storing the word lengths and an integer n representing the total number of lines.

You may assume that the size of the vector is no more than 1,000,000 and all numbers in the vector are less than 1000. (So no word is longer than 999 letters.) In addition, you may assume n is positive and no greater than the size of the vector. **Please don't add any extra functions. Do all of your work inside of this function.**

```
int minLenNeeded(vector<int>& wordLen, int n) {

    int sum = wordLen.size()-1;
    int longWord = 0;
    for (int i=0; i<wordLen.size(); i++) {
        sum += wordLen[i];
        longWord = max(longWord, wordLen[i]);
    }
    int low = longWord, high = sum;           // 5 pts setting low, high

    while (low<high) {                         // 2 pts

        int mid = (low+high)/2;               // 2 pts

        int numLines = 0, curLen = 0;         // 8 pts simulation
        for (int i=0; i<wordLen.size(); i++) {

            int nextLen = curLen + wordLen[i];
            if (curLen > 0) nextLen++;

            if (nextLen > mid) {
                numLines++;
                curLen = wordLen[i];
            }
            else
                curLen = nextLen;
        }
        if (curLen>0) numLines++;

        if (numLines <= n)                    // 2 pt
            high = mid;                       // 2 pt
        else                                  // 1 pt
            low = mid+1;                       // 2 pts
    }

    return low;                               // 1 pt
}
```

7) (10 pts) Shown below is the storage of an undirected unweighted graph with 10 vertices. $g[i]$ stores the list of vertices directly connected to vertex i by an edge. Here is the full storage of g :

$g[0] = \{2, 6\}$
 $g[1] = \{3, 7, 8\}$
 $g[2] = \{0, 4, 8, 9\}$
 $g[3] = \{1\}$
 $g[4] = \{2, 9\}$
 $g[5] = \{6, 8\}$
 $g[6] = \{0, 5, 8\}$
 $g[7] = \{1\}$
 $g[8] = \{1, 2, 5, 6\}$
 $g[9] = \{2, 4\}$

Show the order in which each vertex gets visited when a breadth first search is run starting at vertex 0. To make sure there is only one correct answer, when enqueueing new vertices into the queue, enqueue them in numeric order. (So, when you enqueue 0's neighbors at the first step, you have to enqueue 2 before you enqueue 6.)

0, 2, 6, 4, 8, 9, 5, 1, 3, 7

8) (5 pts) The nickname of Ghana's World Cup team is the Black Stars. What symbol appears in black on Ghana's flag?

Star (Give to All)