SI@UCF Computer Science Camp
Intro to Competitive Programming in C++
Quiz #2 Solutions
Date: 6/25/2025

1) (20 pts) Complete the program below so that it solves the following problem:

The first line of input contains two positive integers, ***n*** ($2 \le n \le 200{,}000$), and ***t*** ($3 \le t \le 2\text{x}10^9$). The following ***n*** lines contain one positive integer each. Each of these integers is distinct and in between 1 and 10^9. Output "Yes" if there exist two different integers on the list that add up to ***t*** exactly, and output, "No", if no two integers in the list exist that add up to ***t*** exactly. To get significant credit, your solution must run in O(nlgn) time. (This means you can't do a double for loop through the data.) **Do not use a set or map in your solution. (This will be follow up question.)**

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

int main() {

    int n, t;
    cin >> n >> t;
    vector<int> vals(n);
    for (int i=0; i<n; i++)
        cin >> vals[i];

    sort(vals.begin(), vals.end());              // 5 pts

    string res = "No";                           // 3 pts
    int i = 0, j = n-1;

    while (i < j) {                              // 2 pts

        if (vals[i]+vals[j] < t)                 // 3 pts
            i++;

        else if (vals[i]+vals[j] > t)            // 3 pts
            j--;

        else {                                   // 3 pts
            res = "Yes";
            break;
        }
    }

    cout << res << endl;                         // 1 pt


    return 0;
}
```

Grading Note: max 10/20 for O(n^2) solution

2) (15 pts) Alter the solution in question #1 so that instead of sorting the input values (this is what you should have done in the first step…) you avoid sorting the input values and use a set instead. In doing this, make sure you account for the fact that the two numbers that have to add up to the target have to be distinct. Write the code for this solution below. (Just the part that goes in the PUT SOLUTION HERE portion of the code on the previous page.)

```
set<int> nums;                              // 5 pts add to set
for (int x: vals)
    nums.insert(x);
string res = "No";                          // 1 pt init

for (int i=0; i<n; i++) {                   // 1 pt
    int need = t - vals[i];                 // 2 pts
    if (need == vals[i]) continue;          // 2 pts
    if (nums.count(need) > 0)               // 3 pts
        res = "Yes";
}

cout << res << endl;                        // 1 pt
```

3) (10 pts) Jerrod is writing a program that reads in votes for an election and needs to output the results sorted in order by number of votes, breaking ties by name. His idea is to use two maps, one map<string,int> and one map<int,string>. The first map would map each name to the number of votes. He would read in all the input data and build this map. Then, he would essentially copy that data into the second map, making the key the number of votes and the associated value the name. Unfortunately, he got a wrong answer when submitting his solution to this problem. Fundamentally, what is his error?

It's possible that two people will get the same number of votes. In a map, it's not possible for one key to map to two different values. So, if Alia and Simon both get 6 votes, the initial map would show Alia → 6, Simon → 6, but the flipped map can not store 6 → Alia, Simon. Instead, the flipped map would erase over the first entry for six votes with the second entry for 6 votes, which is essentially removing a candidate from the election without due process =)

Grading: up to grader discretion

4) (25 pts) In the game of minesweeper, the game board is a 2D grid where each square is either an underscore ('_') or a star ('*'). The stars represent bombs and the underscores represent valid squares. We define a safe valid square to be a valid square that has 2 or fewer adjacent bombs. A bomb is adjacent to a square if it is next to it either up, down, left, right or in one of the four possible diagonal directions. (Thus, the maximum number of bombs that could be adjacent to a valid square is 8.) Write a function that takes in the game board for minesweeper (a vector of strings, where all characters are either '_' or '*'), and returns the number of the valid squares on that grid that are safe. **Please define your own DX/DY arrays before the function.**

```
const vector<int> DX = { -1,-1,-1,0,0,1,1,1 };         // 4 pts for DX/DY

const vector<int> DY = { -1,0,1,-1,1,-1,0,1 };

int numSafeValid(vector<string>& grid) {

    int res = 0;                                       // 1 pt
    for (int i=0; i<grid.size(); i++) {                // 1 pt
        for (int j=0; j<grid[0].size(); j++) {         // 1 pt

            if (grid[i][j] == '*') continue;           // 2 pts
            int bombs = 0;                             // 1 pt
            for (int z=0; z<DX.size(); z++) {          // 1 pt

                int newX = i + DX[z];                  // 2 pts
                int newY = j + DY[z];                  // 2 pts

                if (newX < 0 || newY < 0 || newX >= grid.size() ||
                    newY >= grid[0].size())            // 4 pts
                    continue;

                if (grid[newX][newY] == '*') bombs++; // 3 pts
            }

            if (bombs <= 2) res++;                     // 2 pts
        }
    }

    return res;                                        // 1 pt
}
```

5) (25 pts) Complete the program below so that it prints out each possible 3 x 3 Magic Square which store the integers 1, 2, 3, 4, 5, 6, 7, 8 and 9. A Magic Square is one such that each row, column and diagonal add to the same value. **In writing your code, do NOT use the fact that this sum has to be 15. Instead, use the first row of the filled in square as the desired target. Note: the permutation array will store integers 0 through 8, so instead, we'll check this for a different target but when we print the square, we'll add 1 to each item in the permutation. Finally note that this solution does NOT use any backtracking.**

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

// Indexes of each row, column and diagonal of a 3 x 3 matrix
// stored in a 1D vector.
const vector<vector<int>> LISTS = {{0,1,2}, {3,4,5}, {6,7,8}, {0,3,6},
                                   {1,4,7}, {2,5,8}, {0,4,8}, {2,4,6}};
const int S = 3;
const int N = S*S;
void go(vector<int>& perm, vector<bool>& used, int k);
bool valid(vector<int>& perm);
void print(vector<int>& perm);

int main() {
    vector<int> perm(N);
    vector<bool> used(N, false);
    go(perm, used, 0);
    return 0;
}

void go(vector<int>& perm, vector<bool>& used, int k) {

    if (k == perm.size()) {
        if (valid(perm))
            print(perm);
        return;
    }

    for (int i=0; i<perm.size(); i++) {

        if (used[i]) continue;            // 2 pts
        perm[k] = i;                      // 2 pts
        used[i] = true;                   // 2 pts
        go(perm, used, k+1);              // 2 pts
        used[i] = false;                  // 2 pts

    }
}
```

```
// Returns true iff this permutation is a valid 3 x 3 Magic Square
// Make use of LISTS!!! Returns false otherwise.
bool valid(vector<int>& perm) {

    int sum = 0;                                  // 5 pts for initial sum
    for (int i=0; i<LISTS[0].size(); i++)
        sum += perm[LISTS[0][i]];

    for (int i=1; i<LISTS.size(); i++) {         // 2 pts

        int tmp = 0;                              // 5 pts set I sum
        for (int j=0; j<LISTS[i].size(); j++)
            tmp += perm[LISTS[i][j]];

        if (tmp != sum)                           // 2 pts
            return false;
    }

    return true;                                  // 1 pt
}

void print(vector<int>& perm) {
    for (int i=0; i<S; i++) {
        for (int j=0; j<S; j++)
            cout << perm[S*i+j]+1 << " ";
        cout << endl;
    }
    cout << endl;
}
```

6) (5 pts) What animal appears on the Panda Express logo?

Panda **(Grading: Give to All)**