Group 7

CSCI 313

HW1-- Question 8: Templated Class

To begin our main function, we created and initialized an integer vector. We then used a for loop to fill the vector with the integers from 1 to 10. Next, we declared a second integer vector, q, using the template class. We used the q vector variable to access the printVector function in the template class and print vector nums, which is the vector we first initialized. Next, we created an integer variable that will be used as the sum. This means that we would like the integers in the vector to add up to this target value. We then ask the user to input the value for the sum. Once we have our sum decided, we create an integer vector that stores all the possible outcomes that add up to the sum. We use the findAllSolutions function in order to fill our p_solutions vector with the possible outcomes. Continuing with our main function, we also have an if statement that will output "No possible solutions…" if the p_solution is empty, meaning that there are no number combinations in our original vector that add up to the sum given by the user. Suppose we do find solutions that add up to the sum given by the user and get the results back from our findAllSolution. We will assign p_solutions to solutions and use a for loop to output all the solutions that were found and terminate the program.

For the private part of our template class, we have the findAllSolutionHelper function that takes in a reference to a vector of type T, this time of type integer. In addition, the findAllSolutionHelper function takes in a reference to a vector that contains the vector results variable (that stores all the possible solutions that add up to target sum), the reference vector solution (that stores the numbers we want to check for), the generic type sum (that contains the current sum of all elements in solution vector), and the generic variable target (that is the "sum"

given by user in the main function). A for loop is then used to put each element to the solution vector and all the elements in the vector get added together and assigned to the variable sum. Now, if we get a sum that is equal to the target, this means that a solution is found. Next, we declare a vector pointer ret and the code vector<T>* ret = new vector<T>(solution), meaning that we are assigning everything in the solution vector to results. Moving on, we have results.push_back(*ret) and this indicates that ret is a pointer which is deferenced to become a vector type and the solution that was found gets added to vector results. The following code: "findAllSolutionHelper(nums, results, solution, i+1, sum, target);" is a recursive call that goes through, and tests, all the different possibilities that might add up to our target sum. Every time we finish checking for possibilities, we remove the last element and check if the new combination of numbers works. When we remove an element from the vector, we then need to subtract that element from the sum which is what this following line of code is doing (sum -= nums[i];).

Within our template class we have two other public functions, which are findAllSolution and printVector. The findAllSolution function has a pointer to the vector whose element is also a vector type T and we named that variable results. We also have another variable solution that is a vector of type T and it stores one possible solution. For example, if our target sum is 10, then a possible vector stored inside the solution could be [1, 2, 3, 4]. Proceeding with our findAllSolution function, we encounter a function call on findAllSolutionHelper (explained in the previous paragraph) and if there are any solutions found there, then "return results" will return it back to main. Finally, the printVector function prints the original vector nums we initialized in main.