Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

12.2 The Subset Sum Problem (Part B - For iTunesEntries)

Part B (**required**) - The Subset Sum Problem For iTunesEntries.

Complete the **main()** and other needed code that solves the **subset sum problem** for any **vector** of **iTunesEntries**.

You have to replace the term int with the term **iTunesEntry** in most places in the previous lab program, but don't do this mindlessly - some **int**s remain **int**s. There is a twist, as well: **iTunesEntry** does not support the **cout << someTuneObject** expression which is used in your **Sublist** class (no doubt, in **showSublist()**), so you have to overload the **<<** operator as a global scope function to make this happen. Likewise, in your first solution you will have an expression similar (or identical) to this:

```
    ...  choices[j].getSum() +  dataSet[k-1]  ...
```

This works fine if **dataSet** is a **vector** of **int**s, but if it is a **vector** of **iTunesEntries**, then it will not work; you can't add an **int** (the return type of **getSum()**) to an **iTunesObject**. Or can you? If you overload the **+** operator as a global scope function, you can. Therefore, you'll have to make that adjustment. When you make these adjustments, you should be able to use the modified **main()** and class on **iTunesEntry vectors** to solve the subset sum problem. Here is your main set up:

```
int main()
{
   int target = 0;
   vector<iTunesEntry> dataSet;
   vector<Sublist> choices;
   int k, j, numSets, max, masterSum, arraySize, newSum,
bestSublist;
   bool foundPerfect;

   // read the data
   iTunesEntryReader tunes_input("itunes_file.txt");
   if (tunes_input.readError())
   {
```

```
      cout << "couldn't open " << tunes_input.getFileName()
         << " for input.\n";
      exit(1);
   }

   // time the algorithm ------------------------
   clock_t startTime, stopTime;
   startTime = clock();

   // create a vector of objects for our own use:
   array_size = tunes_input.getNumTunes();
   for (int k = 0; k < array_size; k++)
      dataSet.push_back(tunes_input[k]);

   cin >> target;
   cout << "Target time: " << target << endl;

   // code provided by student
   //
   //

      choices[bestSublist].showSublist();

   // how we determine the time elapsed -------------------
   stopTime = clock();
   // report algorithm time
   cout << "\nAlgorithm Elapsed Time: "
      << (double)(stopTime - startTime)/(double)CLOCKS_PER_SEC
      << " seconds." << endl << endl;

   return 0;
}
```

Here is an example run. Warning - don't use target times over 1000 until you have debugged the program or you may end up waiting.

```
Target time: 3600
Sublist ----------------------------
  sum: 3600
  array[0] = Cowboy Casanova by Carrie Underwood(236) ,   array[1]
= Quitter by
Carrie Underwood(220) ,   array[2] = Russian Roulette by
Rihanna(228) ,   array[
```

```
4] = Monkey Wrench by Foo Fighters(230) ,   array[5] = Pretending
by Eric Clapto
n(283) ,   array[6] = Bad Love by Eric Clapton(308) ,   array[7] =
Everybody's I
n The Mood by Howlin' Wolf(178) ,   array[8] = Well That's All
Right by Howlin'
Wolf(175) ,   array[9] = Samson and Delilah by Reverend Gary
Davis(216) ,   arra
y[11] = Hot Cha by Roy Buchanan(208) ,   array[12] = Green Onions
by Roy Buchana
n(443) ,   array[13] = I'm Just a Prisoner by Janiva Magness(230) ,
array[14]
= You Were Never Mine by Janiva Magness(276) ,   array[15] = Hobo
Blues by John
Lee Hooker(187) ,   array[16] = I Can't Quit You Baby by John Lee
Hooker(182)


Algorithm Elapsed Time: 0.67 seconds.
```

This time, notice how we get a perfect hour's worth of music in a short list of 79 random tunes. This is typical. If you don't get perfect targets most of the time, your algorithm is probably wrong. Please make sure to implement the algorithm presented in our modules so your code will produce exactly the same sublists as in the Auto Grader's solutions.

Part B run output (test cases) requirement

Required Targets: 0, 1200, 3600, 4799, 100000

Other Targets will be checked by the Auto Grader.

376726.1650026.qx3zqy7

LAB ACTIVITY 12.2.1: The Subset Sum Problem (Part B - For iTunesEntries) 0 / 10

Downloadable files

`main.cpp` , `iTunes.cpp` , `iTunes.h` , and `itunes_file.txt`

Download

Current file: **main.cpp** Load default template...

```
1 // ---------------------------------------------------------------------
2 // Assignment #1 The Subset-Sum Problem Part B - iTunes Version
3
```

```
#include <iostream>
#include <string>
#include <vector>
#include "iTunes.h"
#include <time.h>
using namespace std;

// global scope function prototypes ----------------
int operator+(int n, const iTunesEntry & tune);
ostream & operator<<(ostream & out, const iTunesEntry & tune);

// global scope helper
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.cpp** (Your program) →

Program output displayed here

Coding trail of your work What is this?

History of your effort will appear here once you begin working on this zyLab.

Trouble with lab?