**MAKE 1010 and 1020 LAB ALL HAVE SAME ACTIVITIES!**

**CISP 1020 Lab 4**

**Custom Classes, Templates, Binary Search Trees, Recursion**

**General Information**

This application is some of the code that might go into a fitness application. It allows the user to input an activity and the number of minutes spent on that activity then calculates the equivalent number of steps walked. It is the code that might be part of an app for a Fitbit™, for example.

**Detailed Information**

Upon execution, the program reads a file called tracking.bin which is a binary file with participant data in it and puts this data in a binary search tree of Participant objects ordered by participant last name (that’s why the Participant class has an overloaded operator<). **You must create this binary file and put it in the Debug folder AND put it in the Debug folder’s parent folder.** Create this binary file using student data provided in D2L in an Excel spreadsheet.

If the binary input file doesn’t exist, just create an empty binary search tree. **DO NOT END THE PROGRAM.**

After reading the file data into a binary search tree, print the following main menu (shown with a non-error sample input of 1 from the user)

1 Remove participant

2 Add participant

3 Add activity

4 Calculate total miles walked

5 Pre-order print

6 Exit and save

Option> 1

Do not assume the user will enter an integer for the option. If an invalid option is entered, print an error message and reprint the menu. After every option below, reprint the main menu.

All user input below should be case-insensitive. For example, searching on last name arnold should match Arnold.

If option 1 Remove participant, is entered, prompt the user for the participant’s last name and remove that node from the tree if it exists. Otherwise, if the participant doesn’t exist in the tree, print an error message. Example execution:

Last name: arnold

Participant Cindy Arnold, 123.4 total miles walked removed.

If option 2 Add participant is entered, prompt the user for the participant’s last name, search for that name. If the participant already exists in the tree, print an error message. Example execution:

Last name: arnold

Arnold is already a participant.

If the participant isn’t already in the tree, prompt the user for the participant’s first name and height and insert a node with this information in the tree. Don’t forget to set the number of valid Activity objects to 0. Sample execution:

Last name: Baggins

First name: Bilbo

Height in inches: 42.1

If option 3 Add activity is entered, prompt the user for the last name. If the name is not found, print an error message. Sample execution:

Last name: baggins

Baggins is not a participant.

If the participant is found, prompt the user for an activity. We will assume the user knows the activity numbers (not realistic, but if it were an app, we would get a nice drop-down list to choose from). You may assume the user will enter a valid activity number. Sample execution with Bilbo Baggins playing miniature golf for 45 minutes:

Last name: baggins

Activity: 22

Minutes: 45

Added Bilbo Baggins, tennis, 45 minutes = 1.17 miles

If option 4 Calculate total miles walked is entered, traverse the entire tree, summing total miles walked by all participants. Sample execution:

Total miles walked by \*everyone\* = 1234.56!

If option 5 Pre-order print is entered, print the participants in a pre-order traversal of the tree to standard output.

If option 6 Exit and save is entered, open binary file tracking.bin for writing and write the entire tree to the file **in a pre-order traversal.** Make sure you test this.

Assume the distance in feet in one person’s step, *f*, is given by the following formula where *h* is the person’s height in inches.

*f = (0.413 \* h ) /12*

Adapted from <https://www.verywellfit.com/pedometer-step-equivalents-for-exercises-and-activities-3435742>

|  |  |  |
| --- | --- | --- |
| **Activity** | **Steps in 1 minute** | **Code** |
| Assembly Line | 85 | 0 |
| Basketball (shooting baskets) | 136 | 1 |
| Basketball game | 242 | 2 |
| Bicycling under 10mph | 121 | 3 |
| Bicycling | 242 | 4 |
| Cooking | 61 | 5 |
| Dance | 167 | 6 |
| Football | 242 | 7 |
| Hiking | 182 | 8 |
| House cleaning | 91 | 9 |
| Gardening | 99 | 10 |
| Miniature golf | 91 | 11 |
| Racquetball | 212 | 12 |
| Rowing | 212 | 13 |
| Running 6 mph | 303 | 14 |
| Running 7 mph | 348 | 15 |
| Running 8 mph | 409 | 16 |
| Shopping | 70 | 17 |
| Soccer | 212 | 18 |
| Softball | 152 | 19 |
| Stairs | 273 | 20 |
| Swimming laps | 212 | 21 |
| Tennis | 212 | 22 |
| Walking 3mph | 100 | 23 |
| Walking 4mph | 152 | 24 |
| Weight lifting | 121 | 25 |
| Yoga | 76 | 26 |

You must have at least the following files:

* activity.h and activity.cpp that implements an Activity class with private data and public member functions listed below.
  + Some variable/constant to hold activity names, step equivalents and codes. If this (these) variable(s) are in the class, they should be static. Some kind of array(s) might be nice so that ACTIVITIES[BASKETBALL\_SHOOTING] makes sense. In this example, BASKETBALL\_SHOOTING = 1 and ACTIVITIES[BASKETBALL\_SHOOTING] = “basketball”. A constant like

STEPS[BASKETBALL\_SHOOTING] = 136 could also be useful.

* + Activity code (from table above)
  + Minutes done (i.e., 15 minutes)
  + Default constructor
  + Set and get functions for private data that doesn’t set data to nonsensical values. The set function for miles equivalent will need a parameter of height.
  + A function that returns the miles equivalent of the activity (i.e., 15 minutes of tennis = 212\*15 = 3180 steps. Assuming 2.13 feet per step, that’s 3180 \* 2.13 = 6773.4 feet = ~1.28 miles). The function has one parameter, height.
* participant.h and participant.cpp that implements a Participant class with private data and public member functions
  + Last name
  + First name
  + Height in inches
  + Array of 365 Activity objects
  + Number of valid entries in array of Activity objects
  + Set and get functions for private data. You can write the get function for Activity objects with one parameter, an index, and just return the Activity at that index, (i.e., activity[i]).
  + Overloaded operator< that compares the last names of two participants
* node.h – a template class
* binaryTree.h – a template class
* position.h – an iterator template class
* main.cpp

Any class that dynamically allocates memory **MUST** have a copy constructor and an overloaded operator= both of which perform deep copies and a destructor.

**Relevance**

What else could we add? Here are some ideas. Date of activities. More metrics of the participants besides height such as age and weight. A calculation of calories burned. Measured heart rate for each activity.

For the next two weeks, track your activity in an MS Excel spreadsheet using the codes above. For example, below is the beginnings of my own tracking. The first three columns are my first name, last name and height followed by pairs of activity and minutes columns. My first activity, 24, is walking 4 mph (see table of codes above) and I walked 60 minutes.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cindy | Arnold | 62.5 | 24 | 60 | 25 | 45 | 22 | 55 | 26 | 30 |

**Due Date (see D2L).** Copy your entire c1020axxLab4 Codelite workspace folder to directory N:\CSIT\1020P01\handin. This folder is NOT on PS11, but can be accessed from any PC in any lab on any campus. It can NOT be accessed from home, so make sure you manage your time such that you can turn in you lab in time from school.