SOLUTION TO EXERCISE 5 - ANALYSIS PROBLEM

The following documentation provides one solution for Analysis Exercise 5, which provided partial documentation and requested the development of the remaining items below. Note: All italicized text below serves only to clarify the documentation for students, but would not actually appear in the real documentation.

PROBLEM STATEMENT

Determine a bowling team's average score for all bowlers competing in a competition. Repeatedly request and store the score for an individual bowler and accumulate and count each score until the user enters a negative value. Afterwards, calculate and display the team average, rounding the output to the nearest whole number.

VARIABLE LIST

<table>
<thead>
<tr>
<th>IDENTIFIER</th>
<th>DESCRIPTION</th>
<th>DATA TYPE</th>
<th>SOURCE</th>
<th>USAGE</th>
<th>DESTINATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCORE</td>
<td>Individual bowler's score</td>
<td>Integer</td>
<td>Keyboard</td>
<td>Sentinel and for TOTSCORE</td>
<td>---</td>
</tr>
<tr>
<td>QTY</td>
<td>Quantity of bowlers</td>
<td>Integer</td>
<td>Set to 0</td>
<td>Incremented, tested and for TEAMAVG</td>
<td>---</td>
</tr>
<tr>
<td>TOTSCORE</td>
<td>Total of individual scores</td>
<td>Integer</td>
<td>Set to 0</td>
<td>Accumulates SCORE and for TEAMAVG</td>
<td>---</td>
</tr>
<tr>
<td>TEAMAVG</td>
<td>Team average score</td>
<td>Integer</td>
<td>Calculated</td>
<td>---</td>
<td>Screen</td>
</tr>
</tbody>
</table>

SAMPLE SOFTCOPY

(Given in assignment)

ALGORITHM/FLOWCHART

A *flowchart* is a graphic algorithm that helps to illustrate the flow of control of a program as the processor would advance from one step to another, often in simple sequential order, but sometimes following more complex structures. In this case the processor must use its ability to compare and recognize relationships between data such as greater, less than, or equal to, and make a decision about which step(s) to process next. When the order of execution of steps branches backwards to a previous step, then the structure being followed is known as *repetition*. For more information about what each shape below represents, view the web page about Flowcharting Symbols & Guidelines.

(See next page)
The loop in this flowchart was structured using the "pretest" (leading decision) structure. This choice was somewhat arbitrary, since the "posttest (trailing decision) structure could have been used just as effectively. But the choice of the leading decision approach did require the use of a "prime read" just prior to the entrance to the loop and a subsequent "next read" within the body of the loop to maintain good structure.

Note that, even though the flowchart below involves counting with QTY, the control method used by the loop is "sentinel control". This is because QTY is not the "control variable"; SCORE is. The process of counting is part of the body of this loop, but is not involved in controlling its passage and exit.
DESK CHECK

DATA TRACING CHART

Data Tracing Charts are used to document what would be happening in the computer's memory during the execution of the steps described in an algorithm. A column is provided for each variable in your analysis and for each condition being tested (in a diamond shape). An additional column can be added on the left of the table to serve as a reference to steps in your flowchart if you choose to put reference letters next to the related shapes. This was not done in the solution shown below. Note in the chart below that it only relates to steps in the algorithm that effect the memory.

<table>
<thead>
<tr>
<th>SCORE</th>
<th>QTY</th>
<th>TOTSCORE</th>
<th>TEAMAVG</th>
<th>SCORE &gt;= 0</th>
<th>QTY not = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td></td>
<td></td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>103</td>
<td></td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td></td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>253</td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>145</td>
<td></td>
<td></td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>398</td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td></td>
<td></td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>534</td>
<td></td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

133.5
TEST SOFTCOPY

The numerals on the left of the sample output below are there for the analyst's reference only and will not appear on the screen. Bracketed items indicate values entered by the user.

1 BOWLING TEAM AVERAGE PROGRAM
2 ===============================================
3 4 This program will repeatedly request and store the score of
5 an individual bowler and accumulate and count each one until
6 the user enters a negative score. After that, the program will
7 calculate and display the rounded team average.
8 9 Player's score (negative number when done)? [103]
10 Player's score (negative number when done)? [150]
11 Player's score (negative number when done)? [145]
12 Player's score (negative number when done)? [136]
13 Player's score (negative number when done)? [-1]
14 15 Team average: 134

SOURCE CODE FOR PROBLEM 5

The following source code was not assigned for Analysis Assignment 5, but has been included here to provide a complete example of the program's development.

().'/**********************************************************************/
/* Program: a5.cpp - Project 5 - Coding Solution */
***********************************************************************/

#include <iostream>   /* Include header to allow use of console functions */
using namespace std;
#include <iomanip>    /* Include header to allow use of stream manipulators */

int main ()
{
int SCORE,    /* Each individual bowler's score */
    QTY,      /* Quantity of bowlers */
    TOTSCORE; /* Total of indiv. scores */
float TEAMAVG; /* Team Average */

QTY = 0;      /* Initialize player counter */
TOTSCORE = 0; /* Initialize accumulator */

cout << "BOWLING TEAM AVERAGE PROGRAM\n";
cout << "================================\n";

cout << "This program will repeatedly request and store the score of\n";
cout << "an individual bowler and accumulate and count each one until\n";
cout << "the user enters a negative score. After that, the program will\n";
cout << "calculate and display the rounded team average.\n";

cout << "Player's score (negative number when done)? ";
cin >> SCORE;

while (SCORE>=0)
{
    QTY += 1;
    TOTSCORE += SCORE;
    cout << "Player\'s score (negative number when done)? ";
    cin >> SCORE;
}

if (QTY != 0)
{
    TEAMAVG = static_cast<float> (TOTSCORE) / QTY;
    cout << setprecision(0) << fixed << "\nTeam Average: " << TEAMAVG << endl;
}

return 0; /* Send a null error code to the parent process */