Sampe program design for COMP102 Homework # 2

Problem Description:

Write a program for computing the scores for a series of bowling games.

Input specifications:

1. Number of games to be entered by user.
2. Option selected by user on whether the scores are to be entered by user or generated randomly.
3. If user selected to input scores, then a sequence of roll scores will be input.
   The roll scores are assumed to have values between 0 to 10 if it is the first roll in a frame and between 0 to (10 - first roll score) if second roll.

Output specifications:

1. The roll scores will be displayed if they are generated by the random number generator.
2. The total score for each game will be output.
3. The overall total, minimum, average and maximum scores for the series of games will be output.

1.0 Initial algorithm:

4. Ask user for the number of games.
5. Ask user if scores are to be input by user or generated randomly.
6. Calculate and print the score for each game.
7. Update the accumulated total, current minimum and maximum scores.
8. Output the minimum, average, and maximum scores.

2.0 Level-one refinement:

1. Ask user for the number of games.
2. Ask user if scores are to be input by user or generated randomly.
3. Calculate and print the score for each game.

   3.1 For each game, calculate the game score

      3.1.1 Initialize game score to 0
      3.1.2 Get the scores for the first two rolls (R1 & R2) in the game using next_score (at least two scores are needed in computing the score for any frame).
      3.1.3 For each frame, calculate the frame score
      3.1.4 Update current game score

   3.2 Print score for current game.
4. Update the accumulated total, current minimum and maximum scores.
5. Output the minimum, average, and maximum scores.

Refinement for step 3:

3. Calculate and print the score for each game.

3.1 For each game, calculate the game score
    3.1.1 Initialize game score to 0
    3.1.2 Get the scores for the first two rolls (R1 & R2) in the game using next_score (at least two scores are needed in computing the score for any frame).
    3.1.3 For each frame, calculate the frame score
        3.3.1 If a strike is score in the current frame (i.e. R1 equal to 10)
        3.3.2 Calculate score for strike
            3.3.2.1 Get score for the next roll (R3)
            3.3.2.2 Score for this frame is R1 + R2 + R3
        3.3.3 Update the next two scores to be used in the next frame
            3.3.3.1 Set R1 to R2
            3.3.3.2 Set R2 to R3
        3.3.4 If a spare is score in the current frame (i.e. R1 + R2 equal to 10)
        3.3.5 Calculate score for spare
            3.3.5.1 Get score for the next roll using next_score (R3)
            3.3.5.2 Score for this frame is R1 + R2 + R3
        3.3.6 Update the next two scores to be used in the next frame
            3.3.6.1 Set R1 to R3
            3.3.6.2 If this is not the last frame, set R2 to the score for the next roll
        3.3.7 If this is an open frame
        3.3.8 Calculate score for open frame
            3.3.8.1 Score for this frame is R1 + R2
        3.3.9 Update the next two scores to be used in the next frame
            3.3.9.1 If this is not the last frame,
            3.3.9.2 set R1 to the score for the next roll using next_score and
            3.3.9.3 set R2 to the score for the next roll using next_score
    3.1.4 Update current game score
3.2 Print final game score for current game.

Refinement for step 4:

4. Update the accumulated total, current minimum and maximum scores

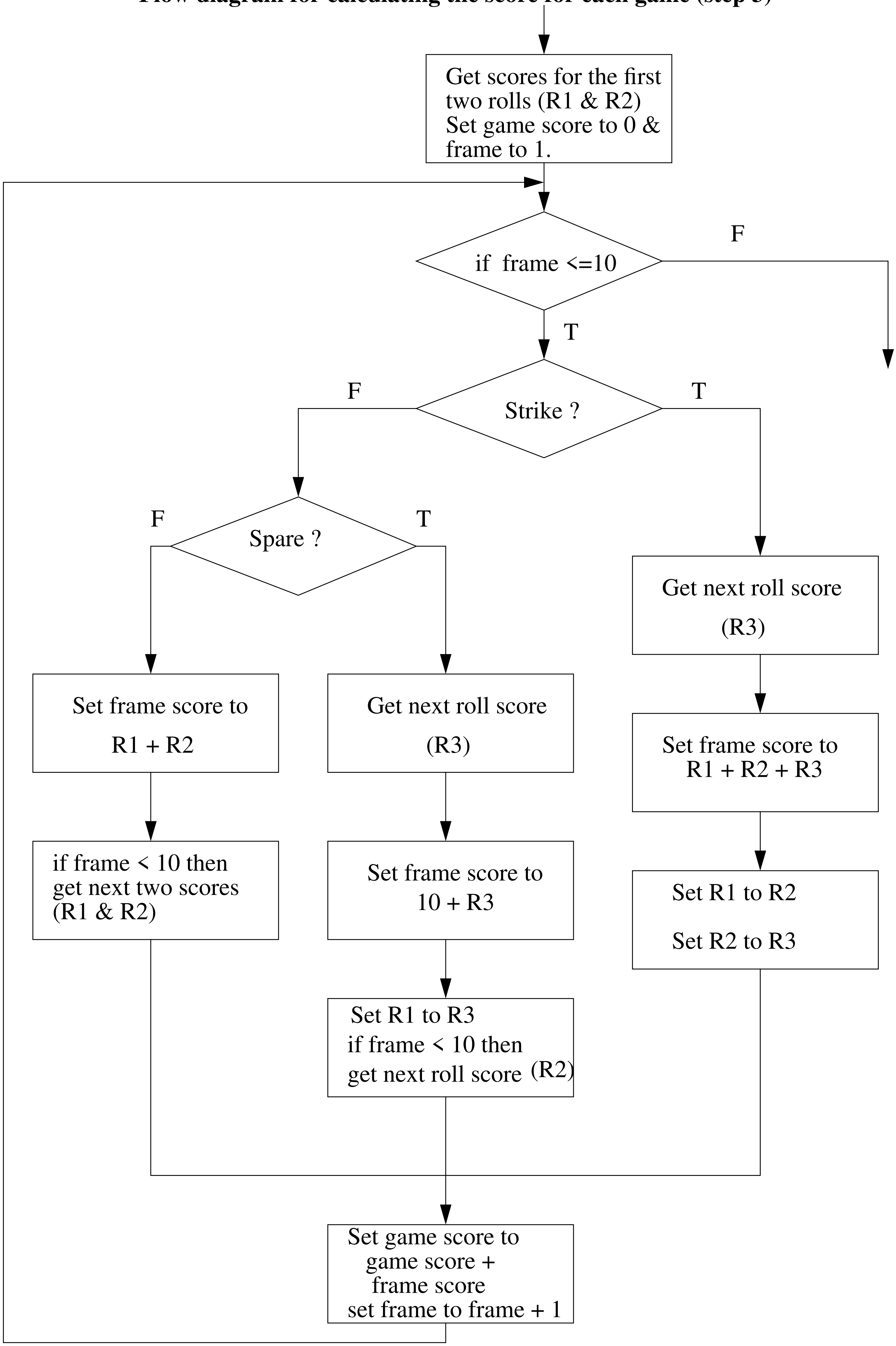
4.1 Add game score to accumulated total.
4.2 if game score < minimum, then set new minimum to game score.
4.3 if game score > maximum, then set new maximun to game score.

Description for next_score:

1. if roll scores are to be input by user
    1.1 ask user to input a score
    1.2 return the input value as the next score

2. if roll scores are to be generated by a random number generator
    2.1 if this is the first roll in a frame
        2.1.1 generate a random number between 0 to 10
        2.12 return the number generated as the next score
    2.2 if this is the second roll in a frame
        2.2.1 generate a random number between 0 to (10 - first roll score)
        2.12 return the number generated as the next score
    2.3 Output the next score generated

Flow diagram for calculating the score for each game (step 3)



```
// Calculates and prints out the game scores for a series of bowling games. Also prints out the
// minimum, average, maximum of all the game scores.

#include <iostream.h>
#include <stdlib.h>

// function prototypes
void framecalc(int, int&, int&, int&, int&);
void gamecalc(int, int&);
void update(int, int&,  int&, int&);
int next_score(int, int, int);

void main() {
  int gamescore, max = 0, min = 300 ,  total = 0 , n;
  int user_random;
  float average;

  cout << "Enter number of games: ";
  cin >> n;

  cout << "Enter 1 (user) or 2 (random): ";
  cin >> user_random;

  for ( int i=1;  i<=n;  i++) {
      gamecalc(user_random, gamescore);
      update(gamescore,  min, max, total);
      cout << endl;
      cout << "Total score for game " << i << " is " << gamescore << endl;
  }

  average = float(total) / n ;
  cout << endl;
  cout << "Overall results for " << n  << " games:" << endl
       << "Total = " << total << endl
       << "Minimum = " << min << endl
       << "Maximum = " << max << endl
       << "Average = " << average << endl;
}

// Generate a score for the next roll in the game. The score can either be entered by the user or
// generated by a random number generator.
int next_score(int user_random, int r1_r2, int r1_score) {
  int nscore;

  if (user_random == 1)
      cin >> nscore;
  else {
      if ( r1_r2 == 1 )
          nscore = random(11);                 // first roll has value between 0 and 10
      else
          nscore = random(11 - r1_score);      // second roll has value between 0 and (11-r1_score)

      cout << nscore << " ";
  }
  return nscore ;
}  // end next_score
```

```
// Calculate the framescore for the current frame, add it to gamescore, increase frame by 1,
// and update rollscore and r1.
void framecalc(int user_random, int& gamescore, int& frame, int& rollscore, int& r1) {
  int framescore, r2;
  if (rollscore == 10) {                                    // The current frame is a strike
    if (r1 == 10)   r2 = next_score(user_random, 1, 0);
    else r2 = next_score(user_random, 2, r1);
    framescore = 10 + r1 + r2;
    rollscore = r1;
    r1 = r2;
  }
  else {
    if ((rollscore + r1) == 10) {                           // The current frame is a spare
      r2 = next_score(user_random, 1, 0);
      framescore = 10 + r2;
      rollscore = r2;
      if (frame < 10)                                       // Get another roll score if not last frame
        if (rollscore == 10)   r1 = next_score(user_random, 1, 0);
        else   r1 = next_score(user_random, 2, r2);
    }
    else {                                                  // This is an open frame
      framescore = rollscore + r1;
       if (frame < 10) {                                    // Get next two roll scores if not last frame
        rollscore = next_score(user_random, 1, 0);
        if (rollscore == 10)   r1 = next_score(user_random, 1, 0);
        else   r1 = next_score(user_random, 2, rollscore);
      }
    }
  }
  gamescore = gamescore + framescore;
} // end framecalc

// Calculate gamescore for the current game
void gamecalc(int user_random, int& gamescore) {
  int frame, rollscore, r1;
  gamescore = 0;
  cout << "Roll scores: " << endl ;
  rollscore = next_score(user_random, 1, 0);
  if (rollscore == 10)   r1 = next_score(user_random, 1, 0);
  else   r1 = next_score(user_random, 2, rollscore);
  for  ( frame = 1; frame <= 10; frame++ )
      framecalc (user_random, gamescore, frame, rollscore, r1);
} // end gamecalc

// Update the minimun, maximum and average of all the game scores
void update(int gamescore, int& min, int& max, int& total) {
  total = total + gamescore;
  if (gamescore < min)
    min = gamescore;
  if (gamescore > max)
    max = gamescore;
}  // end update
```