The code used to simulate games 1 and 2 is listed here. It is written in a language called Perl. For details on reading Perl, see:

http://www.perl.com/pub/q/documentation

The block beginning with sub game explains how games 1 and 2 are played. The block beginning with while enables the program to see data about students’ chosen price ranges. The first block beginning with foreach simulates playing the game 10,000 times for each price acceptable to each player. The second foreach block ranks the students by their accounts. Observe that the code does not directly reward theoretically optimal play.

#!/usr/local/bin/perl
# Everyone plays a fixed number of games at each price they accept.
$GAME = 2;
$I = 10000;
$DBG = 1;

# Random variable equal to the value of a single roll
# The parameter is a random number in [0,1).
sub game {
    if($GAME==1) {
        if($_[0]<3/9) {
            return 0;
        } elsif($_[0]<6/9) {
            return 12;
        } elsif($_[0]<8/9) {
            return 63;
        } else {
            return 100;
        }
    } elsif($GAME==2) {
        if($_[0]<1/10) {
            return 0;
        } elsif($_[0]<3/10) {
            return 5;
        } elsif($_[0]<6/10) {
            return 9;
        } else {
            return 100;
        }
    } else { die "No game specified\n"; }
}

# Read player buy-in data from stdin
while(<>) {
    @s = split;
    $buyin{$s[0]} = $s[1];
}

# Play the game
foreach $k (keys %buyin) {
    print $k, " playing at buy-in ", $buyin{$k}, "\n" if $DBG>1;
    for($price=0; $price<=$buyin{$k}; ++$price) {
        print "price ", $price, "...\n" if $DBG>1;
        for($i=0; $i<$I; ++$i) {
            $net = game(rand()) - $price;
            print "...wins ", $net, "\n" if $DBG>1;
            $account{$k} += $net;
            print "...balance= ", $account{$k}, "\n" if $DBG>1;
        }
    }
}

# Display the results
foreach $k (sort { $account{$b} <=> $account{$a} } keys %account) {
    print $k, "\t";
    printf '% 15.2f', $account{$k};
    print "\n";
}