

Test Sheet 4 (dice duel)

by Connor Johnson

Submission date: 09-Jan-2026 01:19PM (UTC+0000)

Submission ID: 272674843

File name: 65019_Connor_Johnson_Test_Sheet_4__dice_duel__703116_1824663492.docx (257.96K)

Word count: 1659

Character count: 8693

Name: Connor Johnson

Student Number: 2517511

WORKSHEET 4 – TESTING (Dice Duel)

1. Original code:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <Windows.h>

void main()

{
    int myRand[3];
    int compRand[3];
    int high = 6, low = 1;
    int myTotal, compTotal, myScore = 0, compScore = 0;
    char choice;

    printf("are you ready to roll your dice? (y/n) ");
    scanf(" %c", &choice);

    while (choice == 'Y' || choice == 'y')
    {
        srand(time(NULL));
        for (int i = 0; i < 3; i++)
        {
            myRand[i] = rand() % (high - low + 1) + low;
        }

        printf("you rolled:\n");
        printf("%d %d %d\n", myRand[0], myRand[1], myRand[2]);

        myTotal = myRand[0] + myRand[1] + myRand[2];

        for (int i = 0; i < 3; i++)
        {
            compRand[i] = rand() % (high - low + 1) + low;
        }
        printf("the computer has rolled:\n");
        printf("%d %d %d\n", compRand[0], compRand[1], compRand[2]);
    }
}
```

```

        compTotal = compRand[0] + compRand[1] + compRand[2];

        printf("You rolled %d and the computer rolled %d\n\n", myTotal, compTotal);

        if (myTotal > compTotal)
        {
            printf("You win!");
            system("color 2a");
        }
        else if (compTotal == myTotal)
        {
            printf("draw, no points");
            system("color 6a");
        }
        else
        {
            printf("the computer has won!");
            system("color 4a");
        }

        if (myTotal > compTotal)
        {
            myScore = myScore + 1;
        }
        else if (compTotal > myTotal)
        {
            compScore = compScore + 1;
        }

        printf("\n\n");
        printf("current scores: You:%d  computer:%d\n\n", myScore, compScore);

        printf("\n\n");
        printf("Go again (y/n) ");
        scanf(" %c", &choice);
        system("color 0a");
        printf("\n\n\n");
    }
}

```

2. Code Testing

For each test below (i – ix), run your code and show the results of the scenarios below. Present a screenshot of what happened when the programme ran. Did the programme behave correctly or did it do something unexpected, and if so, why?

Part a)

- i. Computer wins round
- ii. User wins round
- iii. Draw
- iv. "Go again" – enter *y*
- v. "Go again" – enter *n*
- vi. "Go again" – enter *Y*
- vii. "Go again" – enter *b*

Part b)

- viii. Show score/score tracking (depending on how you have implemented your program)
- ix. Show an overall win (however this is achieved eg first to 5, best of 12 etc)

Part c)

x. Analyse your code and write a short paragraph commenting on the efficiency of the original code. Could efficiency be improved and if so, comment on what is currently inefficient?

Test:

- 1) Computer wins round

Screenshot(s) of output:

```
you rolled:
3 2 4

the computer has rolled:
3 5 5

You rolled 9 and the computer rolled 13

the computer has won!

current scores: You:1  computer:1
```

Result: Did the programme behave as expected ie syntactically and logically correct?

In this test the programme behaved as expected, the user had a randomised roll followed by the computer. The programme then displayed the total scores for each player and whether the user or computer has won, in this case the computer had rolled higher, and the programme correctly displayed the scores and that the computer had won the round.

Test:

2) User wins round

Screenshot(s) of output:

```
are you ready to roll your dice? (y/n) y
you rolled:
5 3 4

the computer has rolled:
3 4 1

You rolled 12 and the computer rolled 8

You win!

current scores: You:1 computer:0
```

Result: Did the programme behave as expected ie syntactically and logically correct?

In this test the programme behaved as expected, the user chose to roll three dice, and the values were correctly generated. The computer then rolled three dice. The totals were correctly calculated and displayed them, since the user's total was higher the programme correctly displayed 'you win!'.

Test:

3) Draw

Screenshot(s) of output:

```
are you ready to roll your dice? (y/n) y
you rolled:
1 4 6

the computer has rolled:
3 2 6

You rolled 11 and the computer rolled 11

draw, no points

current scores: You:0 computer:0

Go again (y/n) |
```

Result: Did the programme behave as expected ie syntactically and logically correct?

In this test the programme behaved as expected. Both the user and computer rolled three random dice, and the totals were correctly calculated. As both totals were equal, the programme correctly identified the round as a draw.

Test:

4) "Go again" – enter y

Screenshot(s) of output:

```
are you ready to roll your dice? (y/n) y
you rolled:
1 4 6

the computer has rolled:
3 2 6

You rolled 11 and the computer rolled 11
draw, no points
current scores: You:0 computer:0

Go again (y/n) y

you rolled:
6 4 1

the computer has rolled:
6 1 2

You rolled 11 and the computer rolled 9
You win!
current scores: You:1 computer:0

Go again (y/n) |
```

Result: Did the programme behave as expected ie syntactically and logically correct?

In this test the 'go again' feature worked correctly by successfully prompting the user after each round and responding correctly to the input. When 'y' was entered the programme restarted the game loop, generated a new set of dice rolls and saved the current game scores instead of resetting them. This shows that the loop condition and input handling were implemented correctly, allowing repeated game play until the user chooses to stop.

Test:

5) "Go again" – enter *n*

Screenshot(s) of output:

```
are you ready to roll your dice? (y/n)
y
you rolled:
6 1 3

the computer has rolled:
1 1 3

You rolled 10 and the computer rolled 5
You win!

current scores: You:1  computer:0

Go again (y/n) n
```

Result: Did the programme behave as expected ie syntactically and logically correct?

The programme again worked as expected. When 'n' was entered after the 'go again' prompt the programme correctly exited the game loop without starting a new round. The final scores were displayed for both players before termination. This test shows the programme handled the input correctly and ended the programme cleanly confirming the stop condition worked as intended.

Test:

6) "Go again" – enter Y

Screenshot(s) of output:

```
are you ready to roll your dice? (y/n) y
you rolled:
1 3 1

the computer has rolled:
5 4 6

You rolled 5 and the computer rolled 16
the computer has won!
current scores: You:0 computer:1

Go again (y/n) Y

you rolled:
5 3 6

the computer has rolled:
4 4 2

You rolled 14 and the computer rolled 10
You win!
current scores: You:1 computer:1
```

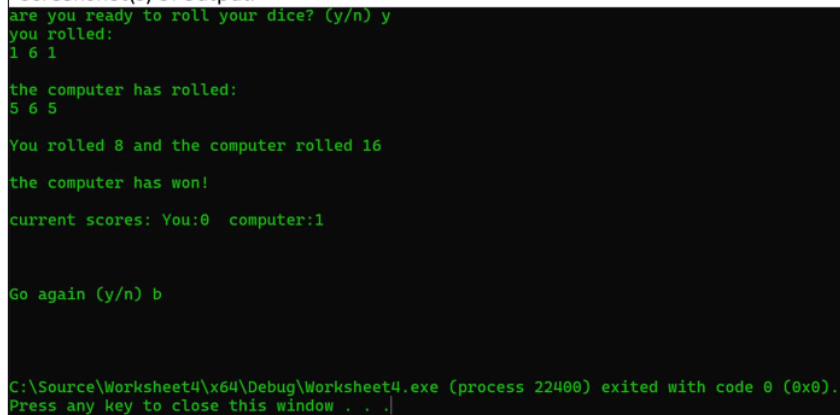
Result: Did the programme behave as expected ie syntactically and logically correct?

In this test the 'go again' feature worked correctly by successfully prompting the user after each round and responding correctly to the input. When 'Y' was entered the programme restarted the game loop, generated a new set of dice rolls and saved the current game scores instead of resetting them. This shows that the loop condition and input handling were implemented correctly, allowing repeated game play until the user chooses to stop.

Test:

7) "Go again" – enter *b*

Screenshot(s) of output:



```
are you ready to roll your dice? (y/n) y
you rolled:
1 6 1

the computer has rolled:
5 6 5

You rolled 8 and the computer rolled 16

the computer has won!

current scores: You:0  computer:1

Go again (y/n) b

C:\Source\Worksheet4\x64\Debug\Worksheet4.exe (process 22400) exited with code 0 (0x0).
Press any key to close this window . . . |
```

Result: Did the programme behave as expected ie syntactically and logically correct?

The programme almost behaved correctly, when 'b' was entered at the go again prompt the input didn't match the valid options of 'y' or 'n', this caused the programme to exit the game loop and end itself cleanly. The previous round still completed normally, and the winner was correctly displayed along with the scores. However, validation could be included here to allow the user to try again explaining the invalid input entered and to give the choice again.

Test:

8) Show score/score tracking (depending on how you have implemented your program)

Screenshot(s) of output:

```
You rolled 11 and the computer rolled 7
You win!
current scores: You:2  computer:2  Go again (y/n) y

Go again (y/n) y                                you rolled:
                                                2 2 1
                                                the computer has rolled:
                                                4 4 4
                                                You rolled 5 and the computer rolled 12
                                                the computer has won!
                                                current scores: You:2  computer:3

                                                Go again (y/n) n
```

Result: Did the programme behave as expected ie syntactically and logically correct?

These code snippets show that some score/score tracking behaved as expected. The output shows that the code successfully tracks scores across each round played and updates and displays the current total after each game. When the user or computer wins a round the corresponding score is increased and carried over to the next round instead of being reset. The updated scores are clearly displayed after each round confirming that the score variables persist correctly inside the programme and provide accurate feedback to the user till the game is ended.

Test:

- 9) Show an overall win (however this is achieved eg first to 5, best of 12 etc)

Screenshot(s) of output:

```
Go again (y/n) y

you rolled:
1 4 1

the computer has rolled:
2 4 4

You rolled 6 and the computer rolled 10

the computer has won!

current scores: You:2  computer:1

Go again (y/n) n
```

Result: Did the programme behave as expected ie syntactically and logically correct?

Within this programme there is no explicit win condition. The game will continue until the user decides to exit. Scores are tracked across rounds, but the game does not automatically end when either the user or computer reaches a specific target such as first to five or best of five. This allows the game to run for any number of rounds be that two rounds or twenty rounds. In a revised version an overall win condition could be added to automatically end the game when a round goal is reached and declare an overall winner.

- 10) Analyse your code and write a short paragraph commenting on the efficiency of the original code. Could efficiency be improved and if so, comment on what is currently inefficient?

The original code is relatively efficient however a few things could be done cleaner. Firstly, looping multiple times to generate three dice for the user and the computer then manually calculating the totals of each roll could be combined into a single loop to save duplication of written code. In a revised version, the random number generator is also written once outside the game loop to avoid the possibility of repeated results. The system colour calls could also be removed as they are unnecessary and add extra complexity while making some outputs harder to read due to the bright colours. Removing these would help clean up both the code and the programme output. Overall, these changes help improve readability, of code and its output, and maintainability.

3. Consider the results of the above tests and your analysis of program efficiency and revisit your original code in order to address any tests that failed and/or improve code efficiency.

(Note: you may not be able to solve all of the problems that were encountered, but solve as many as you can.)

Paste your full revised code here :

[Copy and paste your full revised code here – note, do not use a screenshot]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <Windows.h>

void main()

{
    int myRand[3];
    int compRand[3];
    int high = 6, low = 1, roundCount = 0;
    int myTotal, compTotal, myScore = 0, compScore = 0;
    char choice;

    printf("are you ready to roll your dice? (y/n) ");
    scanf(" %c", &choice);

    printf("\n\n");

    srand(time(NULL));

    while (choice == 'Y' || choice == 'y')
    {
        printf("----- Round %d of 5 ----- \n\n", roundCount + 1);

        myTotal = 0;
        compTotal = 0;

        for (int i = 0; i < 3; i++)
        {
            myRand[i] = rand() % (high - low + 1) + low;
            compRand[i] = rand() % (high - low + 1) + low;

            myTotal += myRand[i];
            compTotal += compRand[i];
        }

        printf("you rolled:\n");
        printf("%d %d %d\n\n", myRand[0], myRand[1], myRand[2]);
    }
}
```

```

printf("the computer has rolled:\n");
printf("%d %d %d\n\n", compRand[0], compRand[1], compRand[2]);

printf("You rolled %d and the computer rolled %d\n\n", myTotal, compTotal);

if (myTotal > compTotal)
{
    printf("You win!\n\n");
}
else if (compTotal == myTotal)
{
    printf("draw, no points\n\n");
}
else
{
    printf("the computer has won!\n\n");
}

if (myTotal > compTotal)
{
    myScore = myScore ++;
}
else if (compTotal > myTotal)
{
    compScore = compScore ++;
}

roundCount++;

if (roundCount == 5)
{
    printf("Game Over!\n\n");
    printf("Final scores: You:%d  Computer:%d\n\n", myScore, compScore);

    if (myScore > compScore)
    {
        printf("You win!\n\n");
    }
    else if (compScore > myScore)
    {
        printf("Computer wins!\n\n");
    }
    else
    {
        printf("The game was a draw!\n\n");
    }
}

```

```
        }  
        break;  
    }  
  
    printf("current scores: You:%d  computer:%d\n\n", myScore, compScore);  
  
    printf("Go again (y/n) ");  
    scanf(" %c", &choice);  
  
    printf("\n\n\n");  
}  
}
```

Test Sheet 4 (dice duel)

GRADEMARK REPORT

FINAL GRADE

16/0

GENERAL COMMENTS

WORKSHEET 4

Coding- The initial code is actually a good shot and it's good to see the use of an array.

Be careful with `srand(time(NULL))` - it should only appear once in a program (I usually put it at the top). Here you've got it in a while loop, so it will be executed more than once - this makes your program less random than it could be.

Good to see some efficiency improvements in the revised code eg using the for loop to add.

Could have included round stats as shown on the Worksheet

Overall a good attempt. Try to keep capital letters at the start of sentences etc in your program output for a more professional feel (just as you would if writing an assignment)

I like the use of colour on the output although I'm not sure green on yellow is readable (but I vaguely remember you telling me you were colour blind? I think??)

Note that in C we just write `myScore++`;

No need for myScore=myScore++;

6.5/10

Testing - Good discussion, please state YES/NO so it's clear to the marker :)

9/10

Overall- 15.5/20

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13
