

NEA sample solution

Task 2 – Cows and bulls (solution 2)

GCSE Computer Science 8520

NEA (8520/CA/CB/CC/CD/CE)

Introduction

The attached NEA sample scenario solution is provided to give teachers an indication of the type of solution that students could complete in response to the specimen material NEA scenario: Task 2 – Cows and Bulls for the new GCSE Computer Science (8520) specification. This specification is for first teaching from September 2016.

This sample solution should only be used to enable teachers to commence planning work for the NEA, (the live NEA scenario will be available for the first time from September 2017). As a result teachers should use this only as a guide to the forthcoming live scenarios. This solution is not a 'real' solution and is provided as an example only. It cannot be used to accurately determine standards in the future.

At our preparing to teach events, spring 2016, the sample scenarios and solutions will be considered and discussed. After these events, appropriate commentaries will be provided, by the senior examining team, to enable teachers to understand the appropriate level achieved by this solution.

Teacher Standardisation events will be used to prepare teachers for the first NEA assessment, which will be available in centres from September 2017. At these meetings teachers will be made aware of the standard.

.
.

Non-exam assessment

Section 1 - Design of Solution

I have been asked to create a program that will play the game Cows and Bulls. This is a game like the board game called Brain Master™ but that game uses coloured pegs rather than numbers.



For this game a 4 digit random number must be created, and the player of the game has to guess the number by entering a guess and getting clues. The clues are the number of bulls and cows. A bull is where the number is correct and in the right place, a cow is where the number is correct but in the wrong place. This is the same as using white or black pegs in the Brain Master game. The game ends when all the numbers are guessed.

TASK 1

The first thing I need to do is to generate 4 random numbers between 0 and 9.

To do this I need to create 4 **integers** and use the random numbers function to generate each number.

```
Random_Number1 <- random number between 0 and 9  
Random_Number2 <- random number between 0 and 9  
Random_Number3 <- random number between 0 and 9  
Random_Number4 <- random number between 0 and 9
```

TASK 2

Next I need to allow the player to enter a 4 digit number that I can check against the random digits generated in task 1.

To do this I will create a function called EnterNumber.

```
input <- EnterNumber()
```

The entry of the user input will need to be in a WHILE loop so that I can keep checking the user input until it is valid. I will use a **Boolean** flag to stop the loop if the input is valid.

```
FUNCTION EnterNumber()

WHILE valid_number = false
  user_input <- USERINPUT
  # validate user_input code goes here
ENDWHILE
```

This would have to ask the user to enter a number between 0001 and 9999. If they happen to enter a number/character that is not a valid option I will output a message telling them to try again as the character they inputted was invalid. I will do this using an IF statement.

I also need to check that the user has not entered "exit". The following code would be in the while loop above.

```
IF user_input == "exit" THEN
  End Program
ENDIF
```

#The user input will be a **string** and I have to validate to see #if it is actually a number / integer. I can use the C# #TryParse method to do this. It tries to convert the string #to an integer.

```
IF user_input is an integer THEN
  valid_number <- true
ELSE
  valid_number <- false
ENDIF
```

#I also need to check that the number is 4 digits long. I will #use the C# Length method.

```
IF LEN(user_input) = 4
  valid_number <- true
ELSE
  valid_number <- false
ENDIF
```

#At the end of the loop I will return the user_input to the #main part of the program.

```
RETURN user_input

ENDFUNCTION
```

TASK 3

Next I need to check the players 4 digit number against the random digits generated in task 1.

```
IF Input_Number1 = Random_Number1 THEN
    NoBulls = NoBulls + 1
ELSEIF
    Input_Number1 = Random_Number2 OR Input_Number1 =
Random_Number3 OR Input_Number1 = Random_Number4 THEN
    NoCows = NoCows + 1
ENDIF

IF Input_Number2 = Random_Number2 THEN
    NoBulls = NoBulls + 1
ELSEIF
    Input_Number2 = Random_Number1 OR Input_Number2 =
Random_Number3 OR Input_Number2 = Random_Number4 THEN
    NoCows = NoCows + 1
ENDIF

IF Input_Number3 = Random_Number3 THEN
    NoBulls = NoBulls + 1
ELSEIF
    Input_Number3 = Random_Number1 OR Input_Number3 =
Random_Number2 OR Input_Number3 = Random_Number4 THEN
    NoCows = NoCows + 1
ENDIF

IF Input_Number4 = Random_Number4 THEN
    NoBulls = NoBulls + 1
ELSEIF
    Input_Number4 = Random_Number1 OR Input_Number4 =
Random_Number2 OR Input_Number4 = Random_Number3 THEN
    NoCows = NoCows + 1
ENDIF
```

#A output is shown saying how many cows and bulls.

```
OUTPUT "You have " + NoBulls + " bulls and " + NoCows + "
cows"
```

TASK 4

Finally the loop ends if there are 4 bulls and so the player has guessed correctly. A count is kept of the number of guesses.

```
IF NoBulls = 4 THEN
    OUTPUT "You have correctly guessed the number in " +
    guesses + " guesses"
ELSEIF
```

Section 2 – Creating the solution

TASK 1

This is the beginning of my code. It gets a random 4 digit number.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace ConsoleApplication1
7 {
8     class Program
9     {
10
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Welcome to Cows and Bulls");
14             Console.WriteLine("a game to guess a number");
15             Console.WriteLine("please enter a 4 digit number or exit");
16
17             //defines all the variable I am using
18             int number1 = 0;
19             int number2 = 0;
20             int number3 = 0;
21             int number4 = 0;
22             int guesses = 0;
23             int noCows = 0;
24             int noBulls = 0;
25
26             Random rnd = new Random();
27             number1 = rnd.Next(0, 10); //generates a random number between 0 and 9
28             number2 = rnd.Next(0, 10); //generates a random number between 0 and 9
29             number3 = rnd.Next(0, 10); //generates a random number between 0 and 9
30             number4 = rnd.Next(0, 10); //generates a random number between 0 and 9
```

TASK 2

This code gets a user number.

```
108 public static string EnterNumber()
109 {
110     //does a loop to check that the user input is valid
111     string input = "";
112     int checknumber = 0;
113     bool validnumber = false;
114     //a loop that stops when the number input is OK
115     while (validnumber == false)
116     {
117         //reads the user input
118         input = Console.ReadLine();
119
120         //if exit is typed then loop breaks
121         if (input == "exit")
122         {
123             break;
124         }
125
126         //converts the input to a number to validate if it is a number
127         if (int.TryParse(input, out checknumber))
128         {
129             validnumber = true;
130         }
131         else
132         {
133             Console.WriteLine("You have entered an invalid number");
134             validnumber = false;
135         }
136
137         // checks the number entered is 4 digits long
138         if (input.Length == 4)
139         {
140             Console.WriteLine("You have entered a valid number");
141             validnumber = true;
142         }
143         else
144         {
145             Console.WriteLine("Your number is too long");
146             validnumber = false;
147         }
148     }
149     return input;
150 }
151 }
152 }
```


TASK 3

This code checks a user number with the random number. A While loop is used, and I had to convert the input to an integer.

```

31         bool NumberGuessed = false;
32
33         //a loop that stops when the correct number is guessed
34         while (NumberGuessed == false)
35         {
36             //calls a function that gets an input from the user and
37             //makes sure it is valid
38             string input = "";
39             input = EnterNumber();
40
41             //if exit is typed in then the program ends
42             if (input == "exit")
43             {
44                 return;
45             }
46
47             //check the entered number against the generated number
48             noBulls = 0;
49             noCows = 0;
50             //splits input number up to check
51             int checknumber1 = int.Parse(input.Substring(0, 1));
52             int checknumber2 = int.Parse(input.Substring(1, 1));
53             int checknumber3 = int.Parse(input.Substring(2, 1));
54             int checknumber4 = int.Parse(input.Substring(3, 1));
55
56             //checks checknumber 1
57             if (checknumber1 == number1)
58             {
59                 noBulls = noBulls + 1;
60             }
61             else if (checknumber1 == number2 || checknumber1 == number3 || checknumber1 == number4)
62             {
63                 noCows = noCows + 1;
64             }
65             //checks checknumber 2
66             if (checknumber2 == number2)
67             {
68                 noBulls = noBulls + 1;
69             }
70             else if (checknumber2 == number1 || checknumber2 == number3 || checknumber2 == number4)
71             {
72                 noCows = noCows + 1;
73             }
74             //checks checknumber 3
75             if (checknumber3 == number3)
76             {
77                 noBulls = noBulls + 1;
78             }
79             else if (checknumber3 == number1 || checknumber3 == number2 || checknumber3 == number4)
80             {
81                 noCows = noCows + 1;
82
83                 }
84             //checks checknumber 4
85             if (checknumber4 == number4)
86             {
87                 noBulls = noBulls + 1;
88             }
89             else if (checknumber4 == number1 || checknumber4 == number2 || checknumber4 == number3)
90             {
91                 noCows = noCows + 1;
92             }

```

TASK 4

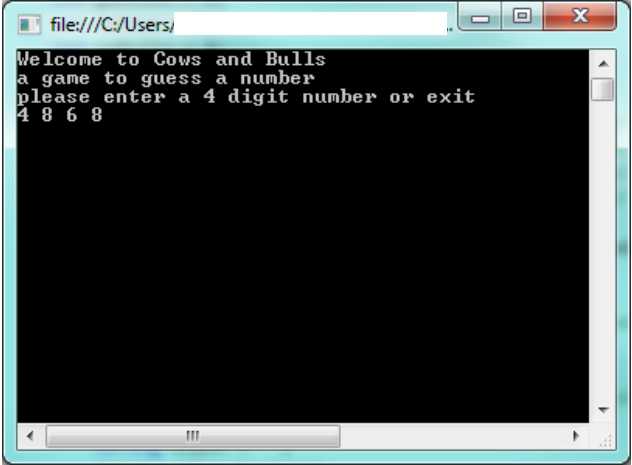
If the user has guessed correctly then a message is displayed.

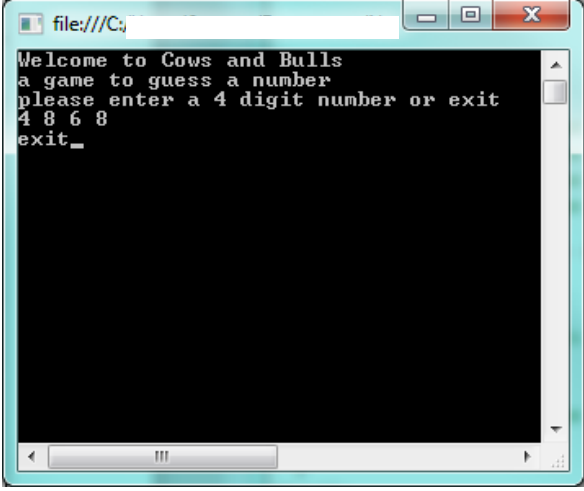
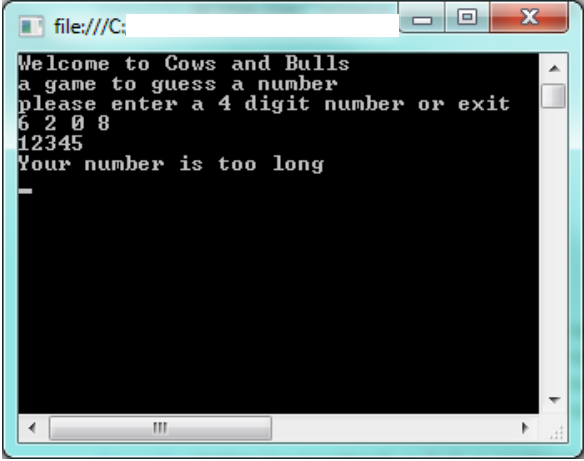
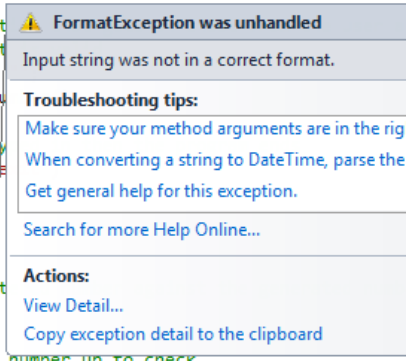
```

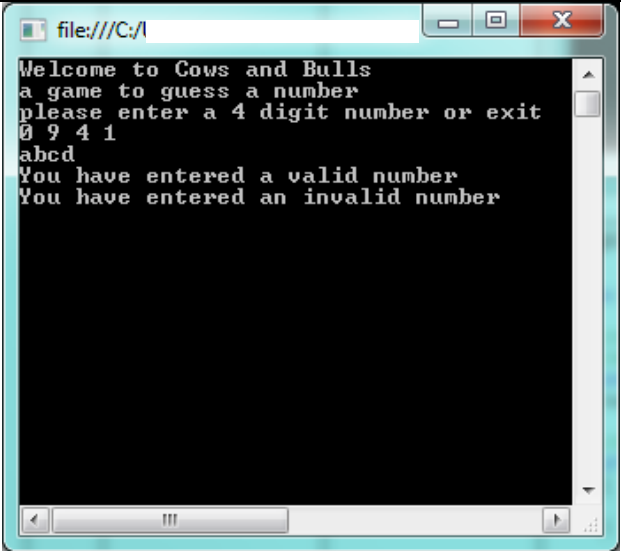
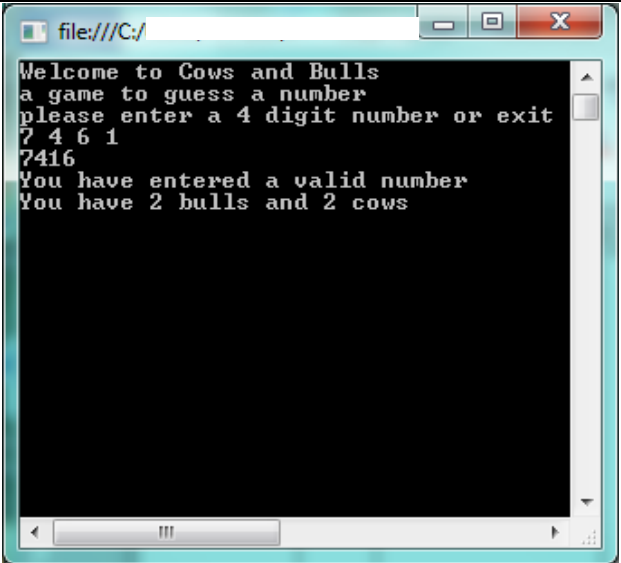
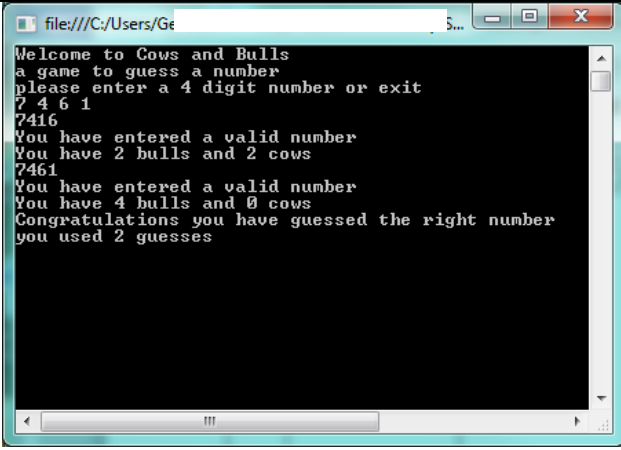
92
93     Console.WriteLine("You have " + noBulls + " bulls" + " and " + noCows + " cows");
94     guesses = guesses + 1;
95
96     //checks if the user has guessed the number
97     if (noBulls == 4)
98     {
99         Console.WriteLine("Congratulations you have guessed the right number");
100        Console.WriteLine("you used " + guesses + " guesses");
101        NumberGuessed = true;
102    }
103 }
104 Console.ReadLine();
105 }

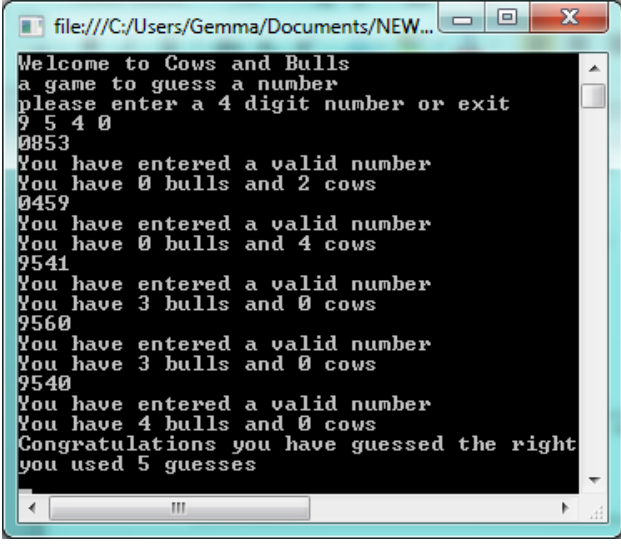
```

Section 3 – Testing the solution

| No. | Description | Expected Outcome | Outcome |
|-----|---|--|---|
| 1 | When I run my code a random number of 4 digits is generated. To prove this I have added a output for testing. | A number is displayed for example 1234 | Worked as expected.  |
| 2 | When I enter exit the program ends | The game will end. | Worked as expected. |

| | | | |
|---|---|---------------------------------------|---|
| | | |  |
| 3 | <p>When a number that is not 4 digits is entered. For example 12345</p> | <p>It will give an error message.</p> | <p>Worked as expected.</p>  |
| 4 | <p>When a number that is not a number is entered. For example abcd</p> | <p>It will give an error message.</p> | <p>I got this error.</p> <pre> while (NumberGuessed == false) { //calls a funct //makes sure it string input = input = EnterNu //if exit is ty if (input == "x") { return; } //check the ent noBulls = 0; noCows = 0; //splits input number up to check int checknumber1 = int.Parse(input.Substring(0, 1)); int checknumber2 = int.Parse(input.Substring(1, 1)); </pre>  <p>I figured it out that the letters were 4 letters long. So I changed my code so the LEN check was done first.</p> |

| | | | |
|---|--|--|---|
| | | |  <pre> file:///C:/ Welcome to Cows and Bulls a game to guess a number please enter a 4 digit number or exit 0 9 4 1 abcd You have entered a valid number You have entered an invalid number </pre> |
| 5 | When a valid number is entered for example 1234 | The correct number of bulls and cows is given |  <pre> file:///C:/ Welcome to Cows and Bulls a game to guess a number please enter a 4 digit number or exit 7 4 6 1 7416 You have entered a valid number You have 2 bulls and 2 cows </pre> |
| 6 | When all the numbers are guessed then a congratulations message is shown | A congratulations message is shown and a count of the guesses. |  <pre> file:///C:/Users/Ge Welcome to Cows and Bulls a game to guess a number please enter a 4 digit number or exit 7 4 6 1 7416 You have entered a valid number You have 2 bulls and 2 cows 7461 You have entered a valid number You have 4 bulls and 0 cows Congratulations you have guessed the right number you used 2 guesses </pre> |

| | | | |
|---|--|-------------|--|
| 7 | Check the count of guesses. For example enter 5 guessed until the number is correct. | Guesses = 5 |  |
|---|--|-------------|--|

Section 4 – Potential enhancements and refinements

Overall, the solution worked appropriately and met the requirements set by the task. It allowed for a 4 digit number to be generated. Furthermore it also let the player enter guesses and gave the number of bulls and cows.

All of the user's input was validated to ensure that the program didn't have any errors that impacted the program running successfully. This was good because it allowed the user to re-enter input and not crash the program. This was done using a function EnterNumber and while loop, so making the solution robust.

To improve it next time I would test whilst developing the program to account for issues that are currently in the program that would affect the program too much. I would also make the program much more efficient and robust by using loops and functions to perform more tasks and minimise the use and creation of new variables.

The big improvement I would make is to add the validation that checks for repeated digits in the random number and the checknumber. I tried to do this but I had too many IF statements and it got confusing. I think a loop would be better but I could not figure out how to do it. Because I had this problem the game was more difficult to play, but was more like the Brain Master game.