

Programming in C

Student Information

Student Name: Tomos Leggett

Student ID: 1901852

Programming in C

Student Information

Exercise 1

Preparation

Problems

Problem #1

Problem #2

Problem #3

Problem #4

Problem #5

Testing

Testing Validation

Code example

Exercise 2

Preparations

Problems

Testing

Code Example

Exercise 3

Preparations

Problems

Problem #1

Problem #2

Problem #3

Problem #4

Testing

Code Example

Exercise 4 (Attempt)

Preparations

Problems

Problem #1

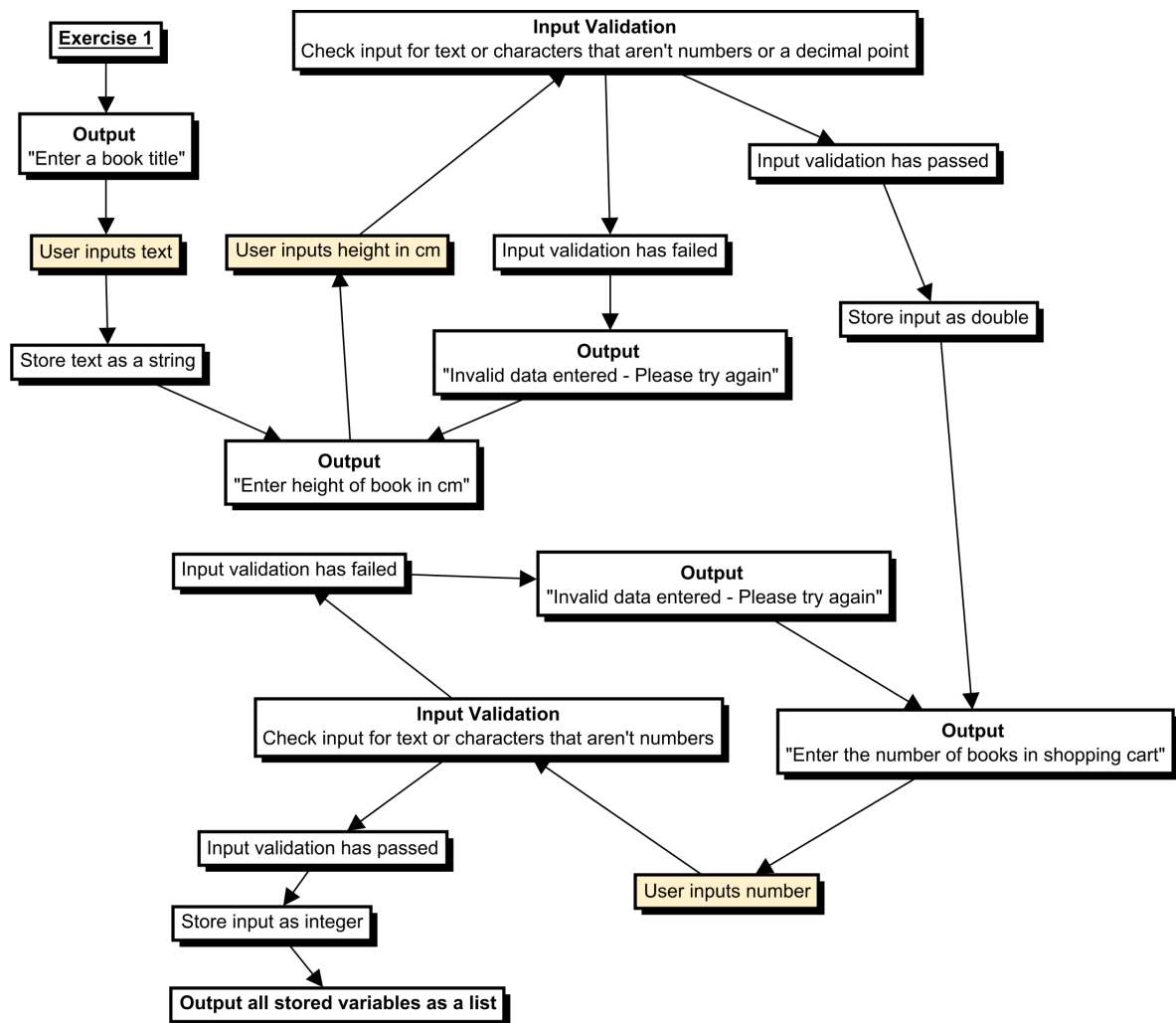
Problem #2

Testing

Code Example

Exercise 1

Preparation



Problems

Problem #1

Whilst working with user input for book height, the author needed a way of validating the data so that it could be stored correctly as a double. If the data wasn't stored correctly, then the program would show an error and allow the user to retry, as opposed to crashing to desktop.

To solidify this issue, the author attempted to implement a system that ensured that only numbers were inserted, and no characters by following the advice from [this stack overflow thread](#) and inserted the following code:

```

1 | std::string s = bookheightUNCONVERTED;
2 |     if (s.find('.') != std::string::npos)
3 |         characterfound == true;
4 |     else
5 |         characterfound = false;

```

Problem #2

The author quickly realised that the `\[` command in C++ represented all characters, including decimal points. To rectify this issue, the author removed the code and followed the advice from [this stack overflow thread](#) instead and inserted the following code:

```

1  try
2      {
3          double bookheight = std::stod(bookheightUNCONVERTED);
4          cout << bookheight << endl;
5      }
6      catch (std::exception & e)
7      {
8          cout << "[!] Invalid entry - Please enter a valid number" <<
endl;
9      }

```

Problem #3

The author noticed another issue that arose in which the program would automatically close if the number was invalid. The author decided to take a different approach and re-run the sub procedure if the data was invalid.

Problem #4

Despite the variable `estimateHeight` being declared here:

```
double estimateHeight = std::stod
```

the developer was confused as to why the variable threw an undeclared error when used as output.

To solidify this, the developer simply declared the variable somewhere else.

```

getline(cin, estimateHeight);
double estimateHeight;
stringstream stream;

try
{
    estimateHeight = std::stod(e
}

```

Problem #5

The method used to ensure validation for using a double does not function when validating an integer. Instead, the developer solidified this issue by researching a conversion solution using [this forum](#) and implementing the following code and using `cin >> {variableName}` instead of `getline(cin, {variableName})`:

```

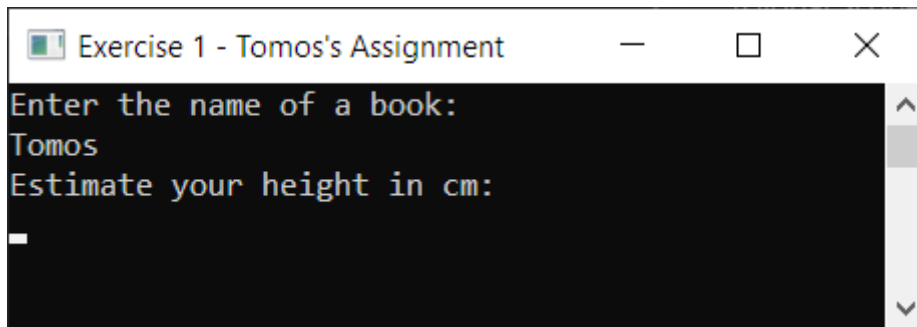
1  while(!cin)
2      {
3          cout << "//ERROR MESSAGE//";
4          cin.clear();
5          cin.ignore();
6          cin >> num;
7      }

```

Testing

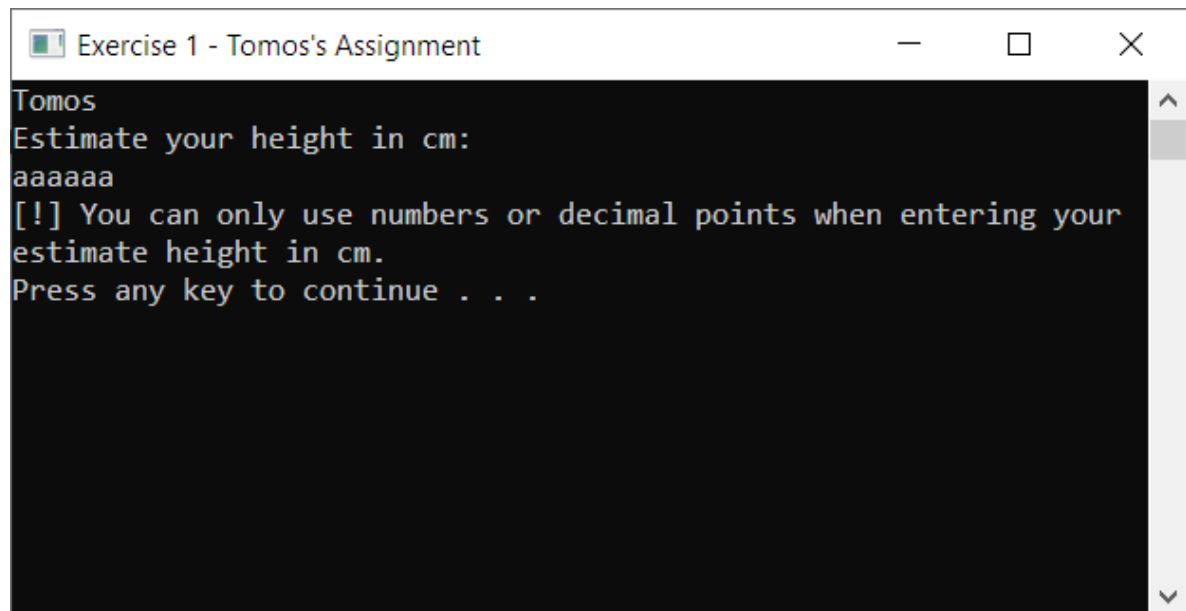
Testing Validation

To ensure proper validation, the developer will now test invalid data types in fields with validation. For example:



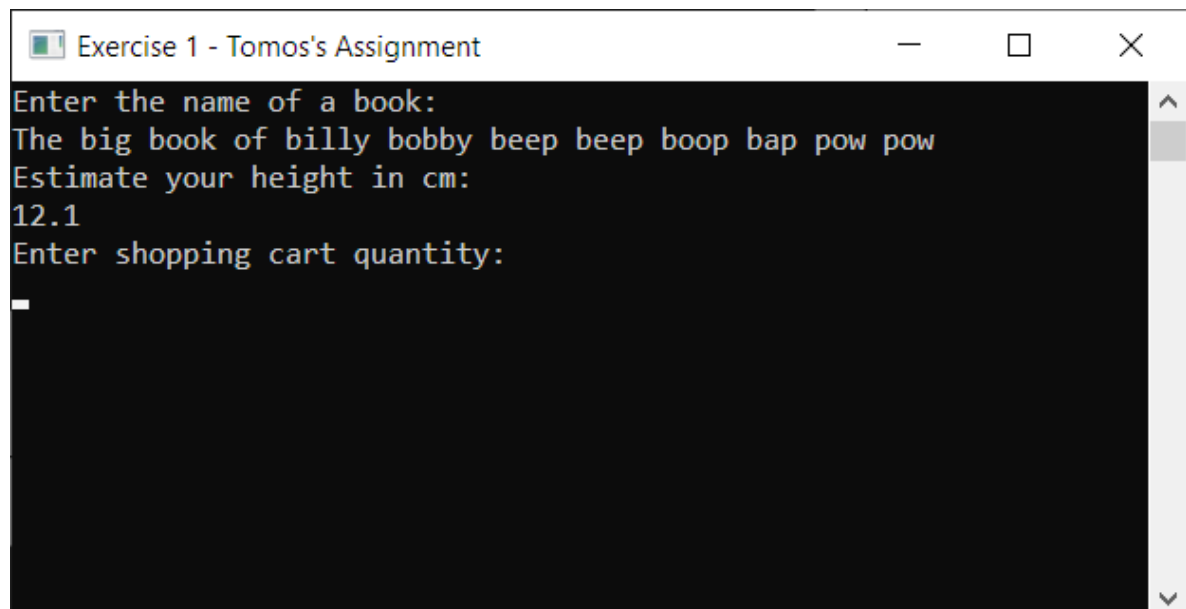
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Tomos
Estimate your height in cm:
_
```

The developer will attempt various different values in this input field. The input field should only accept values that can be stored as a double (numeral values with or without a decimal point) and should not accept characters.



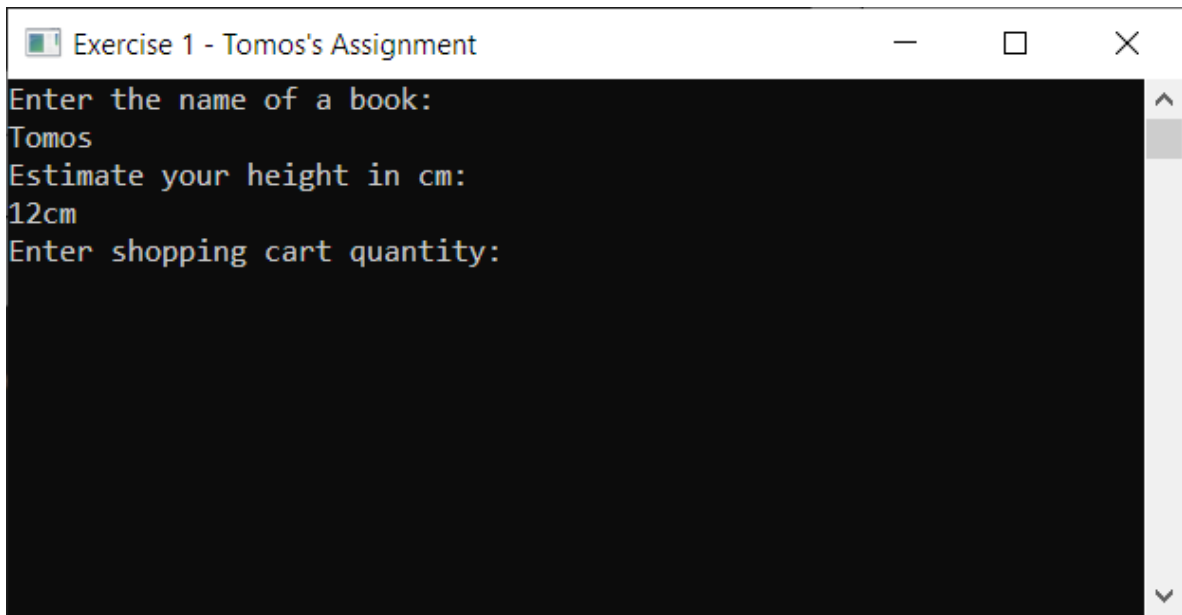
```
Exercise 1 - Tomos's Assignment
Tomos
Estimate your height in cm:
aaaaaa
[!] You can only use numbers or decimal points when entering your
estimate height in cm.
Press any key to continue . . .
```

When entering the value “aaaaaa” the program shows an error and the input restarts



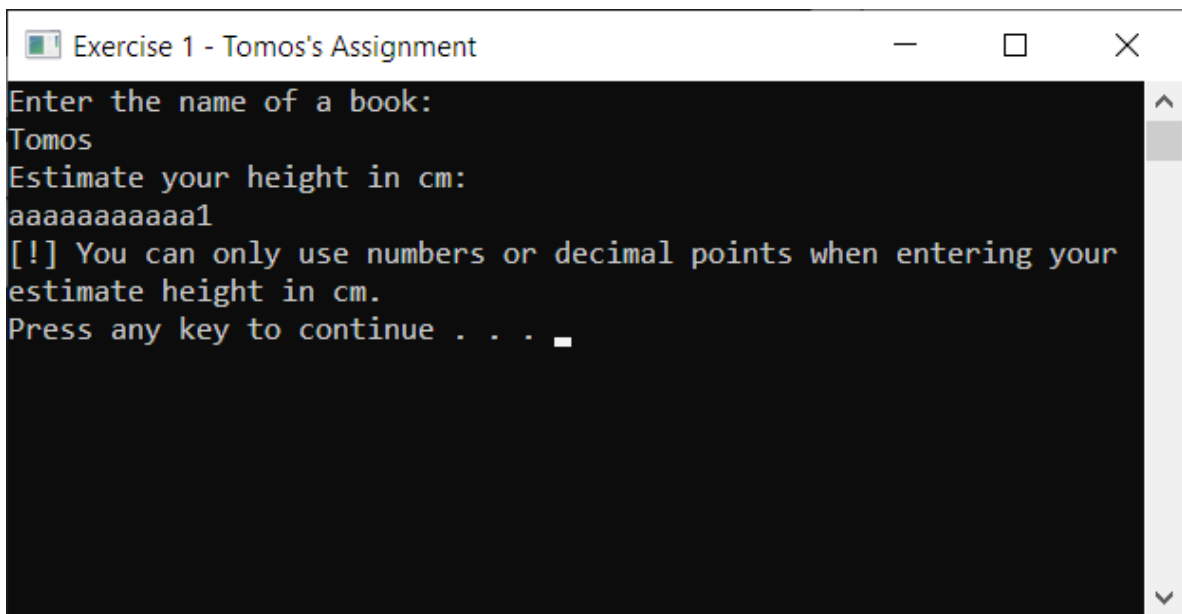
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
The big book of billy bobby beep beep boop bap pow pow
Estimate your height in cm:
12.1
Enter shopping cart quantity:
_
```

When entering the correct value the program moves onto the next input



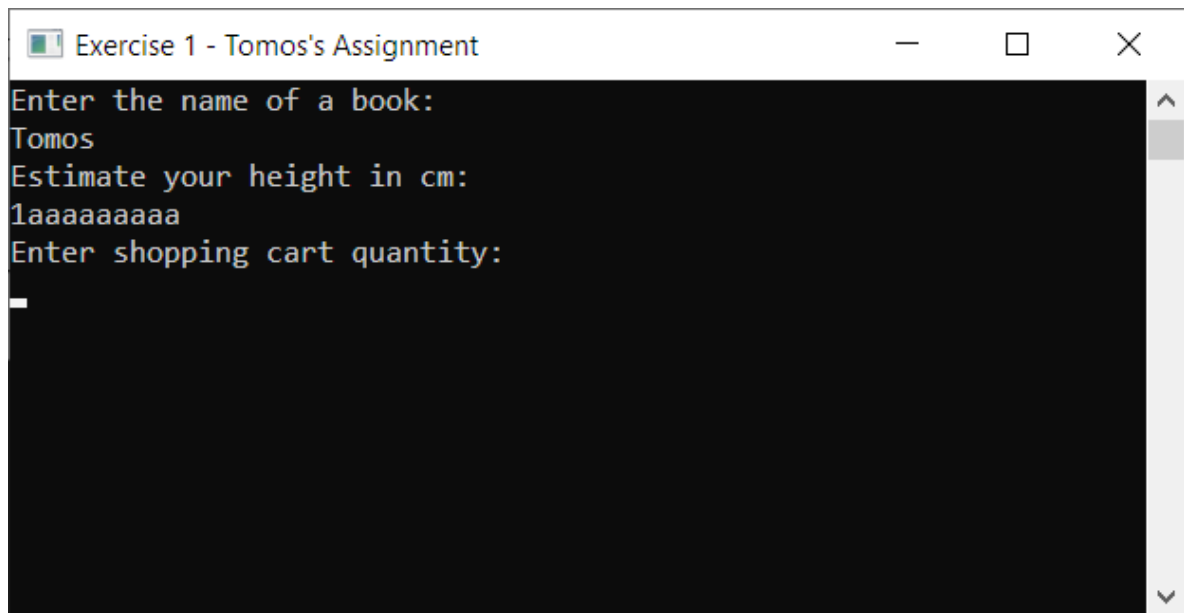
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Tomos
Estimate your height in cm:
12cm
Enter shopping cart quantity:
```

When entering the value “12cm” the program accepts the value, however, it only stores the numeric value, and not the “cm”. This system is put in place to avoid confusion as to whether the user needs to input “cm” at the end of the value



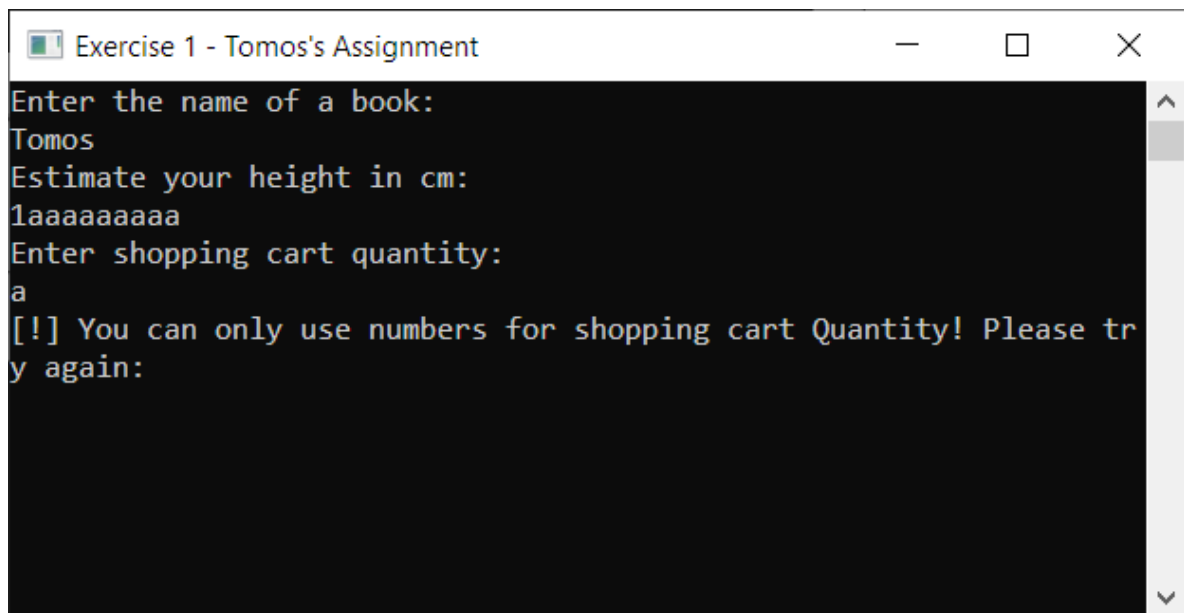
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Tomos
Estimate your height in cm:
aaaaaaaaaa1
[!] You can only use numbers or decimal points when entering your
estimate height in cm.
Press any key to continue . . .
```

When using the value `aaaaaaaaaa1` the program outputs an error and forces the user to input the data again.



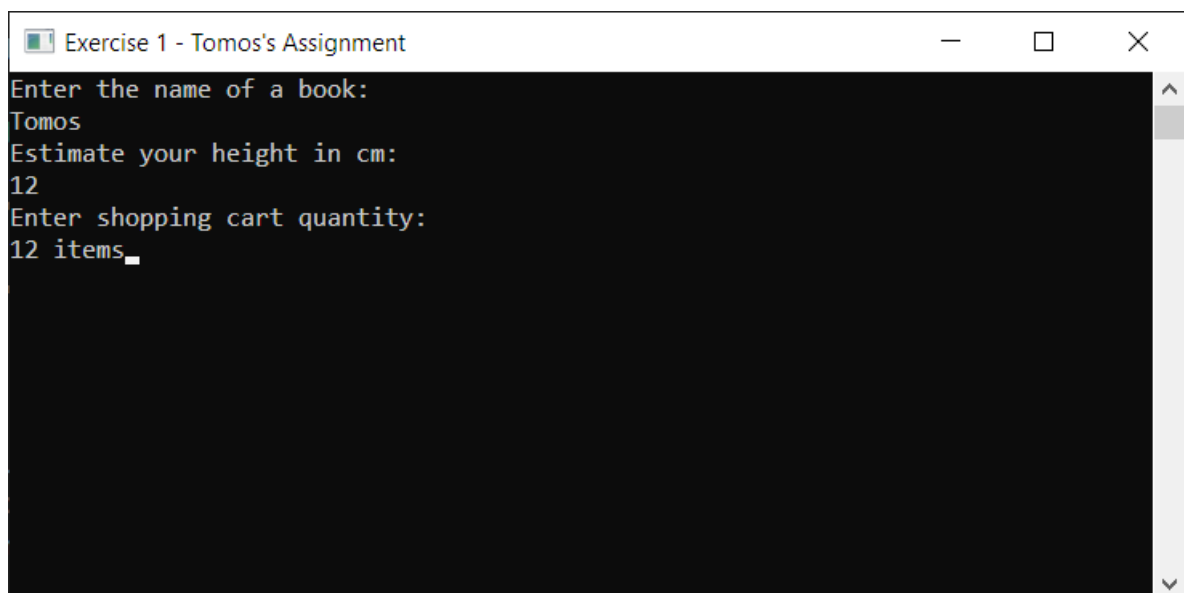
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Tomos
Estimate your height in cm:
1aaaaaaaaa
Enter shopping cart quantity:
_
```

When using the value `1aaaaaaaaa` the program does not output an error and continues onto the next input.



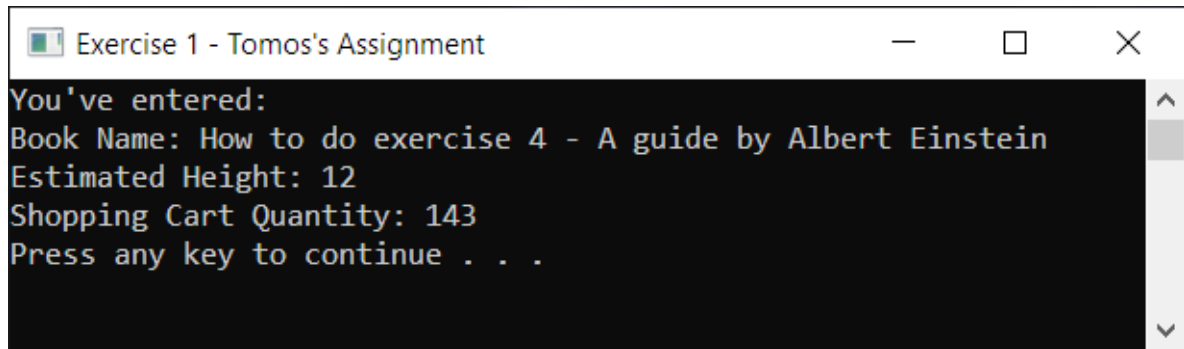
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Tomos
Estimate your height in cm:
1aaaaaaaaa
Enter shopping cart quantity:
a
[!] You can only use numbers for shopping cart Quantity! Please try again:
```

When entering the value “a” when being asked to input the shopping cart quantity, the program throws an error.



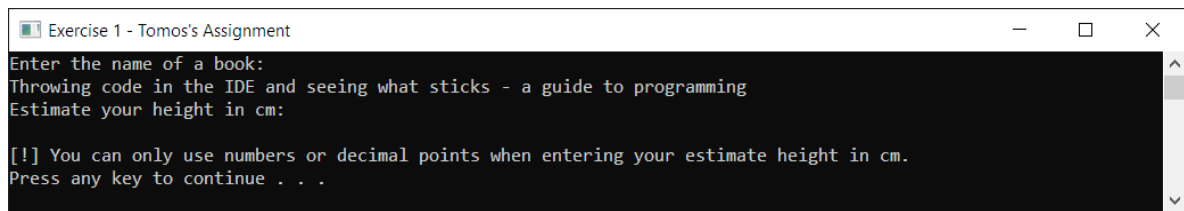
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Tomos
Estimate your height in cm:
12
Enter shopping cart quantity:
12 items_
```

Much like the height input, when the user inputs text after the numerical value, the numerical value is stored and the text is discarded



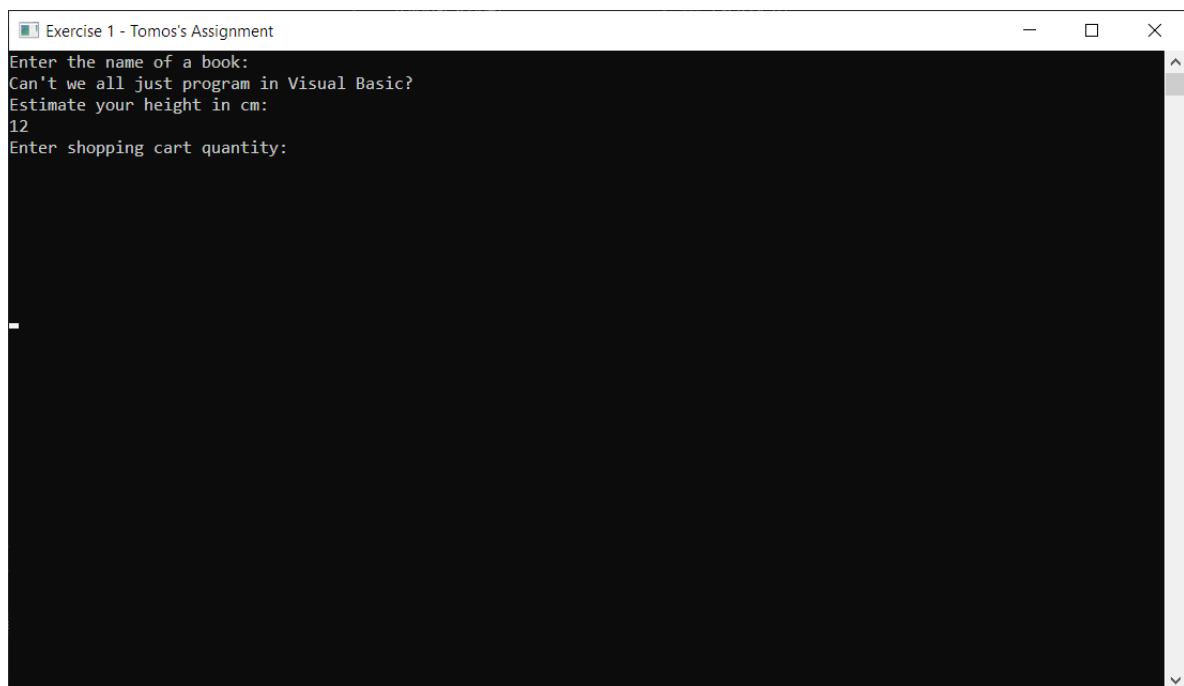
```
Exercise 1 - Tomos's Assignment
You've entered:
Book Name: How to do exercise 4 - A guide by Albert Einstein
Estimated Height: 12
Shopping Cart Quantity: 143
Press any key to continue . . .
```

The program outputs results accordingly, but only when all input value errors are corrected



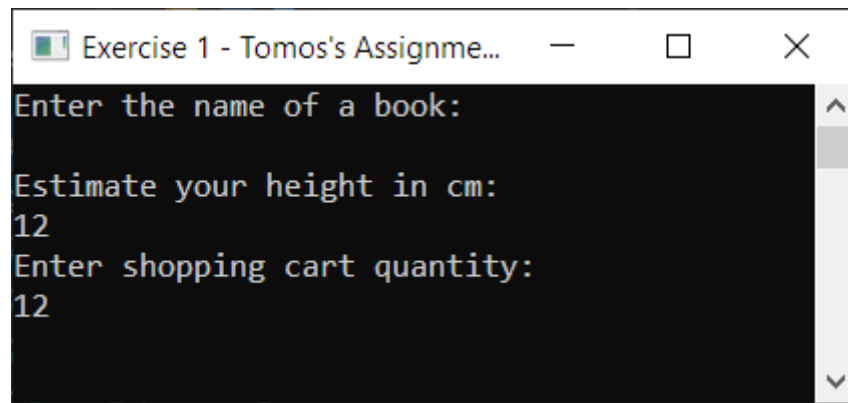
```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Throwing code in the IDE and seeing what sticks - a guide to programming
Estimate your height in cm:
[!] You can only use numbers or decimal points when entering your estimate height in cm.
Press any key to continue . . .
```

When entering no value into height input, an error appears



```
Exercise 1 - Tomos's Assignment
Enter the name of a book:
Can't we all just program in Visual Basic?
Estimate your height in cm:
12
Enter shopping cart quantity:
```

Entering no value on shopping cart quantity input will simply do nothing. The program won't continue unless the user inputs any value.



Entering no value into the book name will store the book name as a blank value.

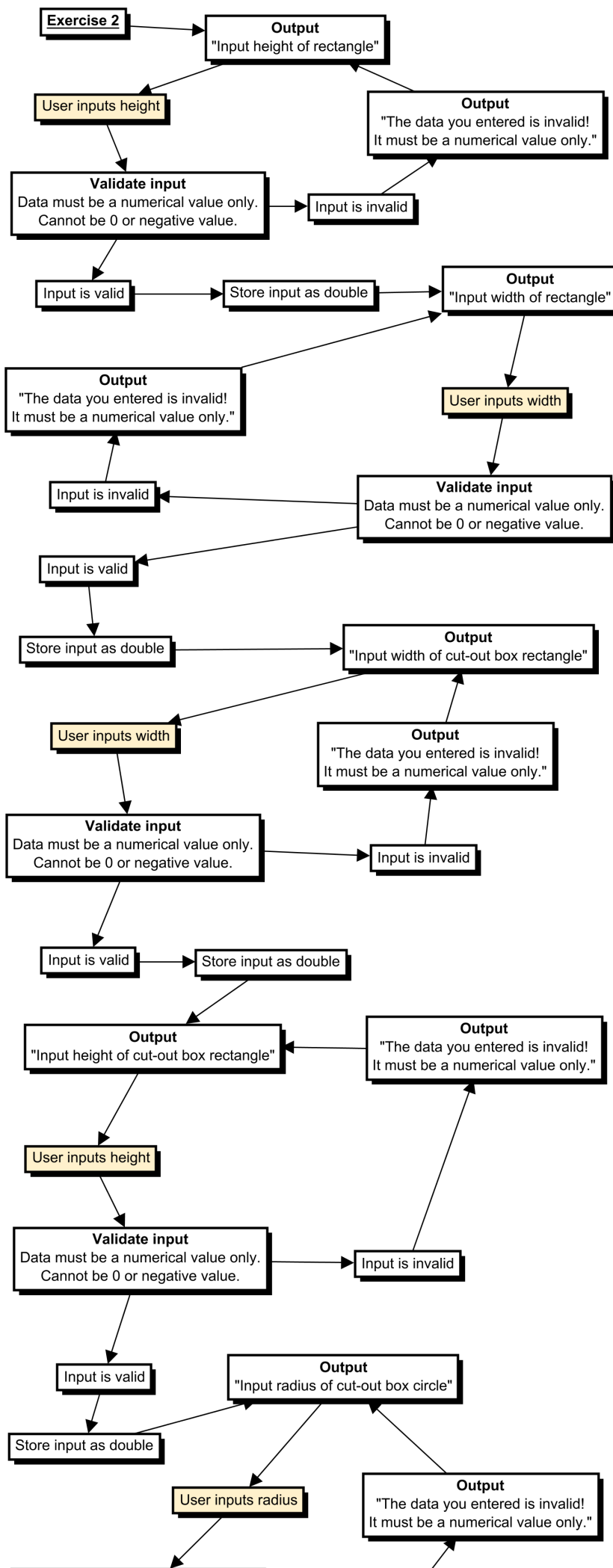
Code example

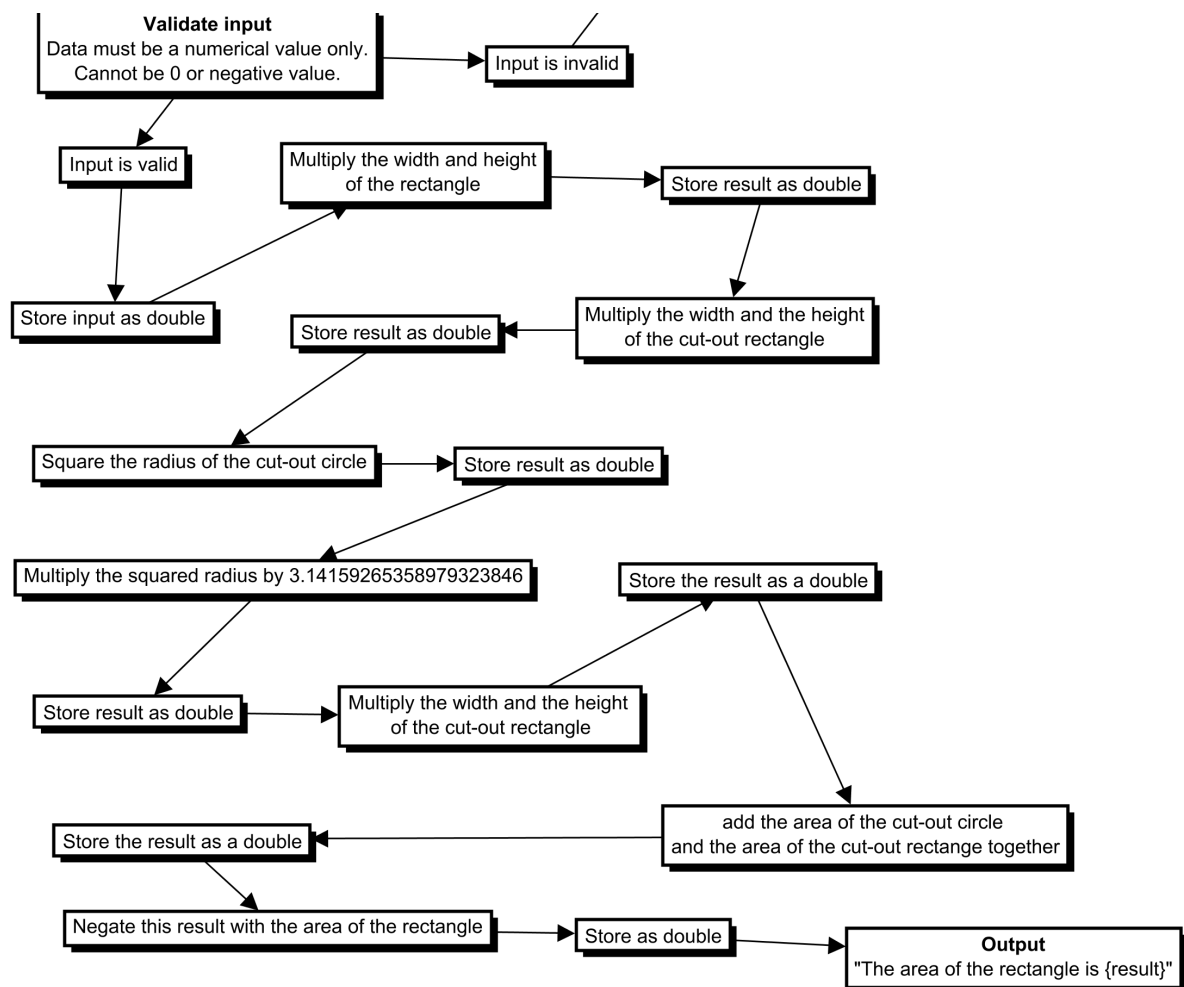
```
1 void exercisel()
2 {
3     system("cls");
4     cout << "Enter the name of a book:" << endl;
5     string bookName;
6     getline(cin, bookName);
7     cout << "Estimate your height in cm:" << endl;
8     string estimateHeightUNC;
9     getline(cin, estimateHeightUNC);
10    double estimateHeight;
11    stringstream stream;
12
13    try
14    {
15        estimateHeight = std::stod(estimateHeightUNC);
16    }
17    catch (std::exception & e)
18    {
19        std::cout << "[!] You can only use numbers or decimal
20    points when entering your estimate height in cm." << std::endl;
21        system("pause");
22        exercisel();
23    }
24
25    int shoppingCartQuantity;
26    cout << "Enter shopping cart quantity:" << endl;
27    cin >> shoppingCartQuantity;
28
29    while (!cin)
30    {
31        cout << "[!] You can only use numbers for shopping cart
32    Quantity! Please try again:" << endl;
33        cin.clear();
34        cin.ignore();
35        cin >> shoppingCartQuantity;
36    }
37
38    system("cls");
39    cout << "You've entered: " << endl;
```

```
38     cout << "Book Name: " << bookName << endl;
39     cout << "Estimated Height: " << estimateHeight << endl;
40     cout << "Shopping Cart Quantity: " << shoppingCartQuantity <<
endl;
41     system("pause");
42     system("cls");
43     mainMenu();
44 }
```

Exercise 2

Preparations





Problems

No problems appeared during the development of this exercise.

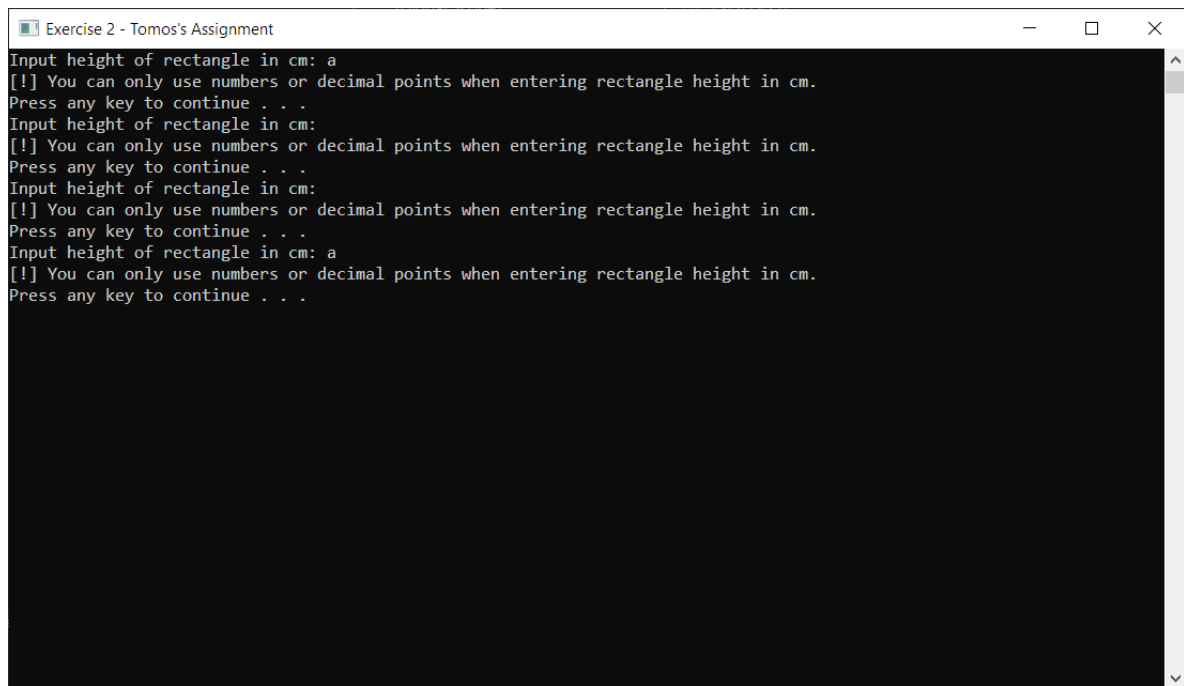
Testing

```

Exercise 2 - Tomos's Assignment
Input height of rectangle in cm:
[!] You can only use numbers or decimal points when entering rectangle height in cm.
Press any key to continue . . .
Input height of rectangle in cm:
[!] You can only use numbers or decimal points when entering rectangle height in cm.
Press any key to continue . . .
Input height of rectangle in cm:
[!] You can only use numbers or decimal points when entering rectangle height in cm.
Press any key to continue . . .
Input height of rectangle in cm:

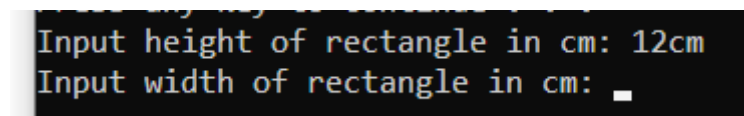
```

Entering no value for the first user input will show an error. The error will not disappear unless an input is given and will loop the same error message every time the user hits enter



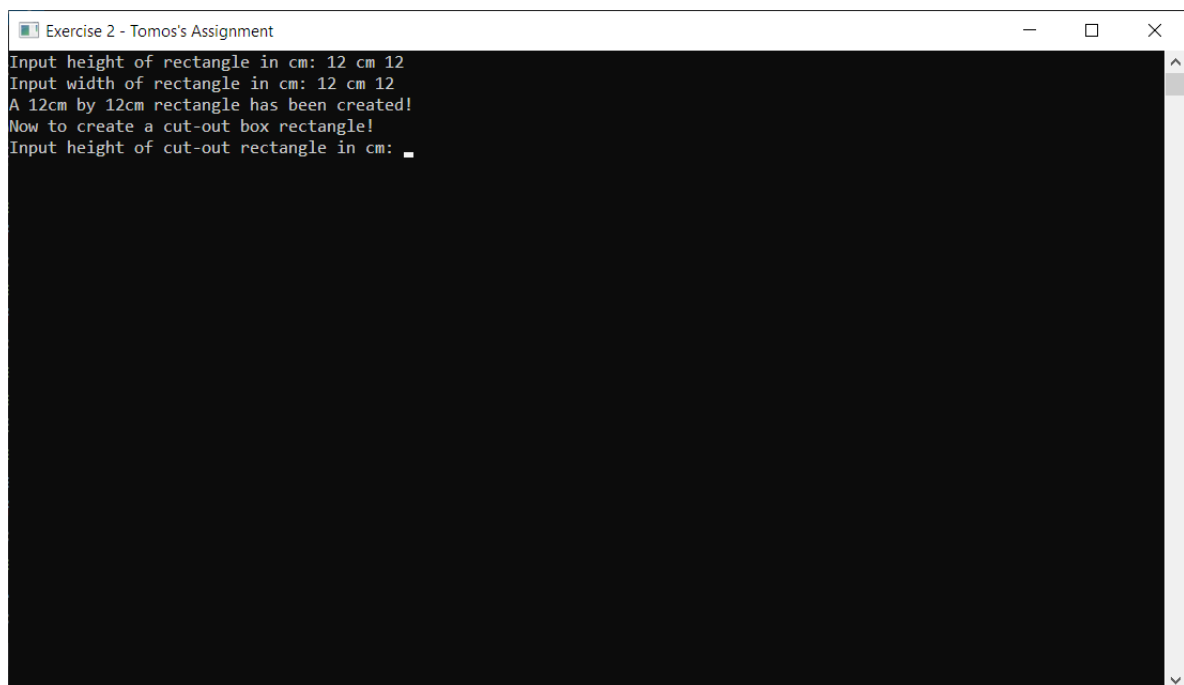
```
Exercise 2 - Tomos's Assignment
Input height of rectangle in cm: a
[!] You can only use numbers or decimal points when entering rectangle height in cm.
Press any key to continue . . .
Input height of rectangle in cm:
[!] You can only use numbers or decimal points when entering rectangle height in cm.
Press any key to continue . . .
Input height of rectangle in cm:
[!] You can only use numbers or decimal points when entering rectangle height in cm.
Press any key to continue . . .
Input height of rectangle in cm: a
[!] You can only use numbers or decimal points when entering rectangle height in cm.
Press any key to continue . . .
```

The same thing happens when no numeric values are entered, such as characters or letters



```
Input height of rectangle in cm: 12cm
Input width of rectangle in cm: _
```

The program will accept text in the input field, but will immediately discard it and store the value only



```
Exercise 2 - Tomos's Assignment
Input height of rectangle in cm: 12 cm 12
Input width of rectangle in cm: 12 cm 12
A 12cm by 12cm rectangle has been created!
Now to create a cut-out box rectangle!
Input height of cut-out rectangle in cm: _
```

When inserting two values, the program will store the first value before a space or a difference

```
Exercise 2 - Tomos's Assignment
Input height of rectangle in cm: 12.1 alalalala
Input width of rectangle in cm: 12.1 alalalalalalalala
A 12.1cm by 12.1cm rectangle has been created!
Now to create a cut-out box rectangle!
Input height of cut-out rectangle in cm: _
```

The program stores double values with decimal points

```
Exercise 2 - Tomos's Assignment
Input height of rectangle in cm: 0
Input width of rectangle in cm: 0
A 0cm by 0cm rectangle has been created!
Now to create a cut-out box rectangle!
Input height of cut-out rectangle in cm: 0
Input width of cut-out rectangle in cm: 0
You've cut out a rectangle of 0cm by 0cm!
Now let's move onto the cut-out circle
Input radius of cut-out circle in cm: 0
The area of the rectangle you entered is: 0cm^2
The area of the cut-out rectangle you entered is: 0cm^2
The area of the cut-out circle you entered is: 0cm^2
Which means if you cut out all the cut-out shapes, the area of the rectangle you entered would be: 0cm^2
Press any key to continue . . .
```

The program will store 0 as a value

Code Example

```
1  system("cls");
2      cout << "Input height of rectangle in cm: ";
3      string rectangleHeightUNC;
4      getline(cin, rectangleHeightUNC);
5      double rectangleHeight;
6      stringstream stream;
7      try
8      {
9          rectangleHeight = std::stod(rectangleHeightUNC);
10     }
11     catch (std::exception & e)
```

```

12     {
13         std::cout << "[!] You can only use numbers or decimal
points when entering rectangle height in cm." << std::endl;
14         system("pause");
15         exercise2();
16     }
17
18     cout << "Input width of rectangle in cm: ";
19     string rectangleWidthUNC;
20     getline(cin, rectangleWidthUNC);
21     double rectangleWidth;
22     try
23     {
24         rectangleWidth = std::stod(rectangleWidthUNC);
25     }
26     catch (std::exception & e)
27     {
28         std::cout << "[!] You can only use numbers or decimal
points when entering rectangle width in cm." << std::endl;
29         system("pause");
30         exercise2();
31     }
32
33     cout << "A " << rectangleHeight << "cm by " << rectangleWidth
<< "cm rectangle has been created!" << endl;
34     cout << "Now to create a cut-out box rectangle!" << endl;
35
36     cout << "Input height of cut-out rectangle in cm: ";
37     string cutoutrectangleHeightUNC;
38     getline(cin, cutoutrectangleHeightUNC);
39     double cutoutrectangleHeight;
40     try
41     {
42         cutoutrectangleHeight =
std::stod(cutoutrectangleHeightUNC);
43     }
44     catch (std::exception & e)
45     {
46         std::cout << "[!] You can only use numbers or decimal
points when entering cut-out rectangle height in cm." <<
std::endl;
47         system("pause");
48         exercise2();
49     }
50
51     cout << "Input width of cut-out rectangle in cm: ";
52     string cutoutrectangleWidthUNC;
53     getline(cin, cutoutrectangleWidthUNC);
54     double cutoutrectangleWidth;
55     try
56     {
57         cutoutrectangleWidth = std::stod(cutoutrectangleWidthUNC);
58     }
59     catch (std::exception & e)
60     {

```

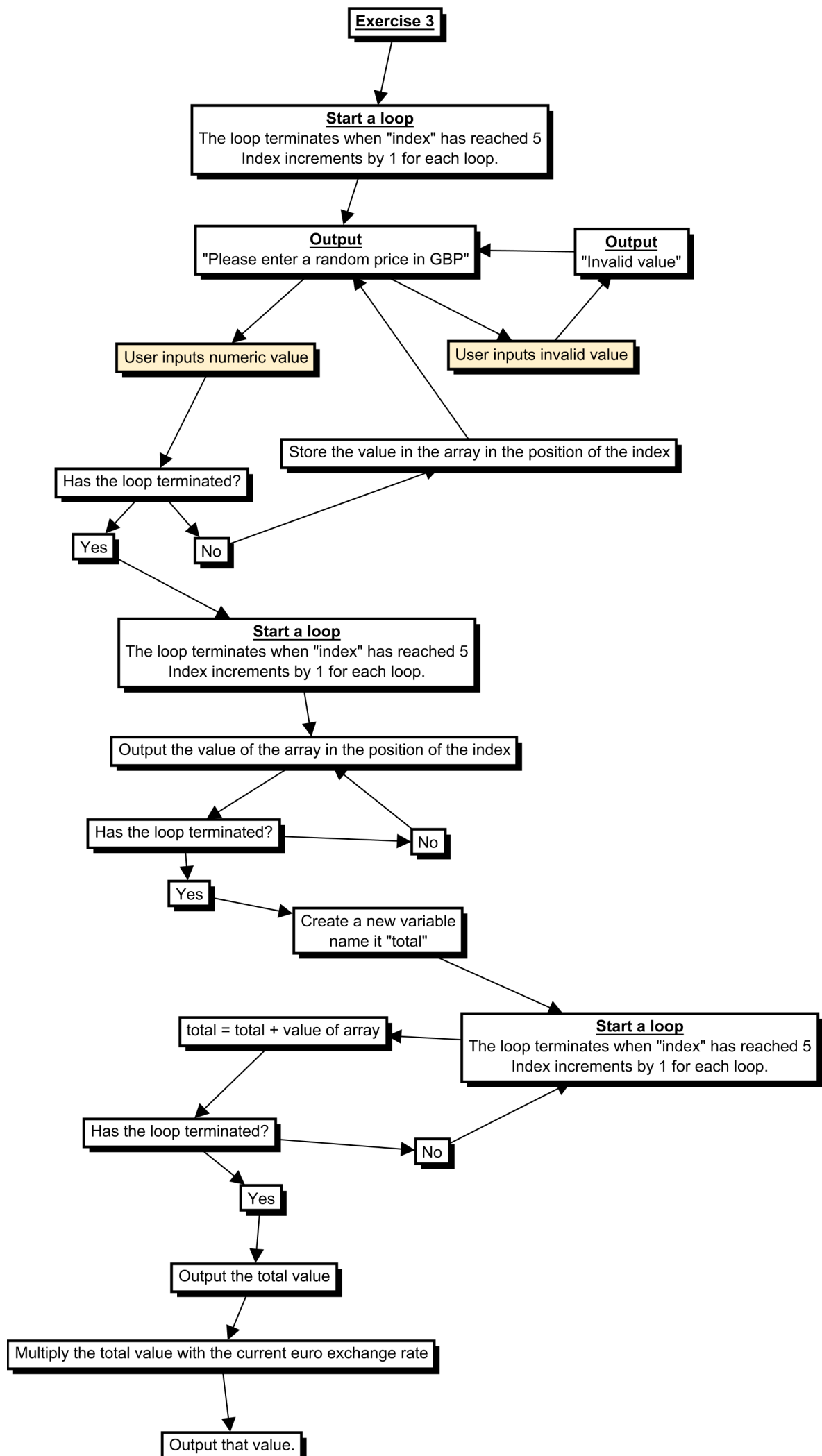
```

61         std::cout << "[!] You can only use numbers or decimal
points when entering cut-out rectangle width in cm." << std::endl;
62         system("pause");
63         exercise2();
64     }
65
66     cout << "You've cut out a rectangle of " <<
cutoutrectangleHeight << "cm by " << cutoutrectangleWidth << "cm!"
<< endl;
67     cout << "Now let's move onto the cut-out circle" << endl;
68
69     cout << "Input radius of cut-out circle in cm: ";
70     string cutoutcircleradiusUNC;
71     getline(cin, cutoutcircleradiusUNC);
72     double cutoutcircleradius;
73     try
74     {
75         cutoutcircleradius = std::stod(cutoutcircleradiusUNC);
76     }
77     catch (std::exception & e)
78     {
79         std::cout << "[!] You can only use numbers or decimal
points when entering cut-out radius of circle in cm." <<
std::endl;
80         system("pause");
81         exercise2();
82     }
83
84     cout << "Let's do some Maths!" << endl;
85     double areaOfRectangle;
86     areaOfRectangle = rectangleHeight * rectangleWidth;
87     cout << "The area of the rectangle you entered is: " <<
areaOfRectangle << "cm^2" << endl;
88     double areaOfCutOutRectangle;
89     areaOfCutOutRectangle = cutoutrectangleHeight *
cutoutrectangleWidth;
90     cout << "The area of the cut-out rectangle you entered is: "
<< areaOfCutOutRectangle << "cm^2" << endl;
91     double areaOfCutOutCircle;
92     double pi;
93     pi = 3.14159265;
94     double radiusSquared;
95     radiusSquared = cutoutcircleradius * cutoutcircleradius;
96     areaOfCutOutCircle = pi * radiusSquared;
97     cout << "The area of the cut-out circle you entered is: " <<
areaOfCutOutCircle << "cm^2" << endl;
98     double concludedAreaOfRectangle;
99     concludedAreaOfRectangle = areaOfRectangle -
(areaOfCutOutRectangle + areaOfCutOutCircle);
100    cout << "Which means if you cut out all the cut-out shapes,
the area of the rectangle you entered would be: " <<
concludedAreaOfRectangle << "cm^2" << endl;
101    system("pause");
102    system("cls");
103    mainMenu();

```

Exercise 3

Preparations



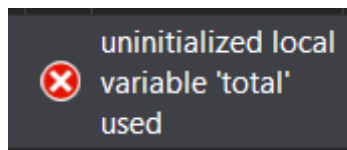
Problems

Problem #1

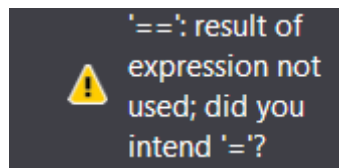
When attempting to use the following code to calculate the total prices:

```
1 double total;
2
3     for (index = 0; index < 5; index++)
4     {
5         total = (total + prices[index]);
6     }
```

The IDE would throw the following error:



Other warnings and issues were declared by Visual Studio, such as this strange error:



The developer wasn't sure how to fix this error, as the code this error was pointing towards didn't use "==" and the error simply wasn't true.

To solidify this issue, the developer followed a suggestion from the built in error solution menu for Visual Studio and inserted "{ }" after the variable "total". As for the error, it seemed to have disappeared after retyping the code.

Problem #2

The pound symbol was displaying the wrong character.

The solution to this problem was to use the C++ command: `char(156)` as opposed to directly using £.

Problem #3

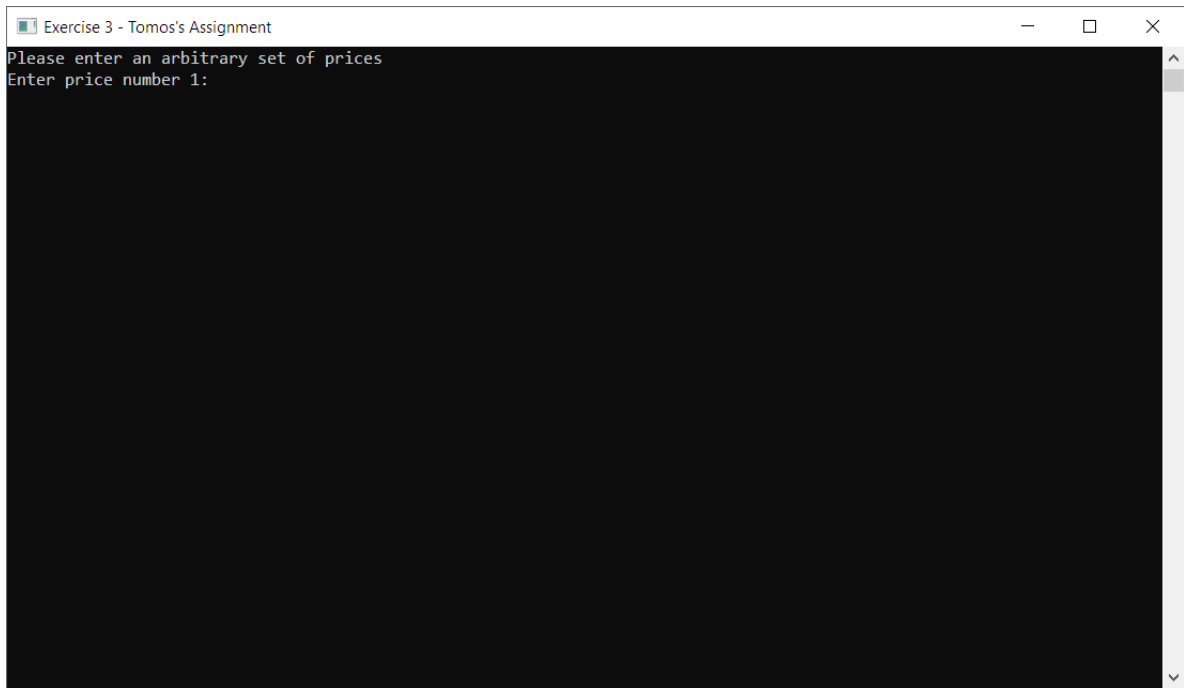
The euro symbol was also displaying the wrong character.

The solution to this problem was to use the C++ command: `\u20AC` as opposed to directly using €

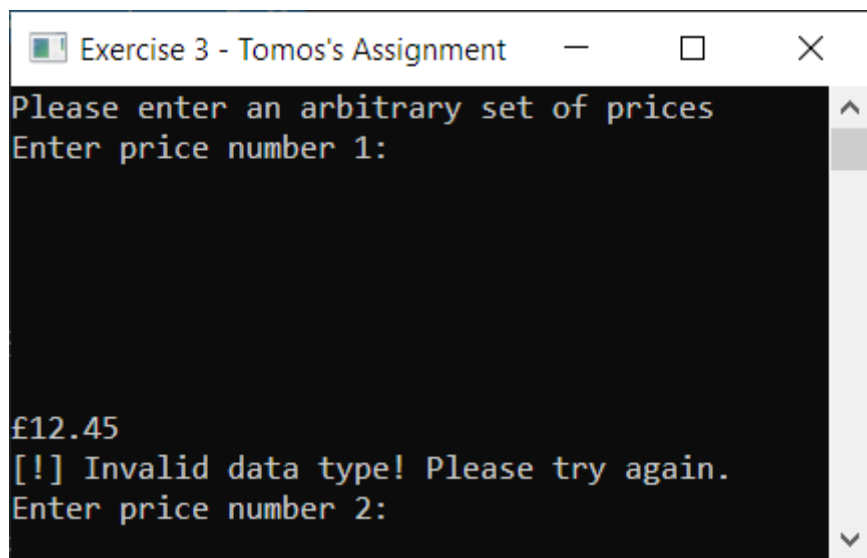
Problem #4

The command `\u20AC` also didn't work, the developer researched a solution and found that the windows command line does not render Unicode characters such as the euro symbol, so the word "euros" was used instead.

Testing



When entering no value, the cursor simply drops down a line. You cannot physically enter no value.



Entering values that contain characters will show an error, however, the program will store the data as a double, discard any characters and continue regardless

```
Exercise 3 - Tomos's Assignment
Please enter an arbitrary set of prices
Enter price number 1: a
[!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
a
Enter price number 1: [!] Invalid data type! Please try again.
1
Enter price number 1: 1
Enter price number 2:
```

Using invalid characters will show an error, the error will disappear once a valid value has been entered, but does not store that value unless it is entered again once prompted to enter the value for some reason?

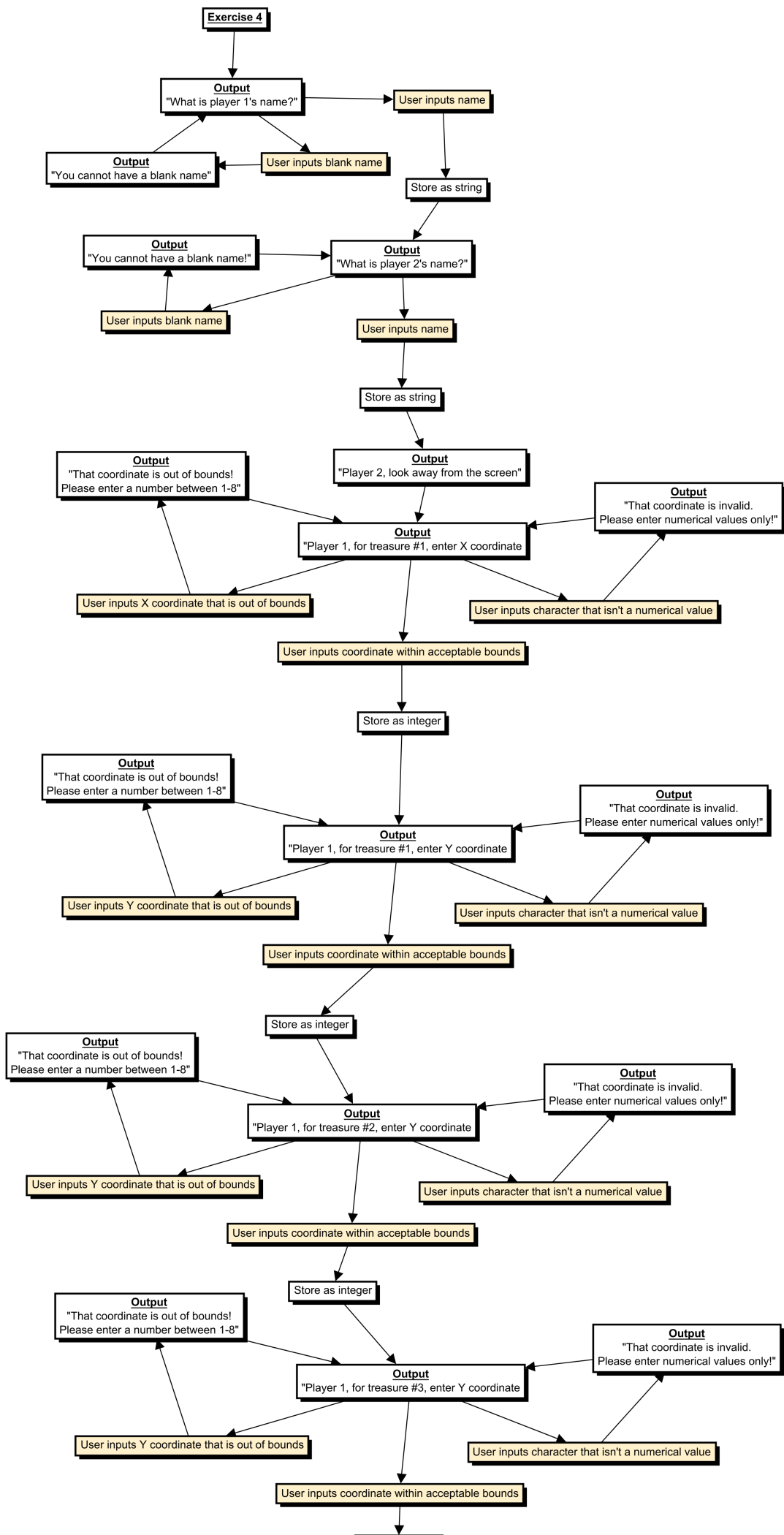
Code Example

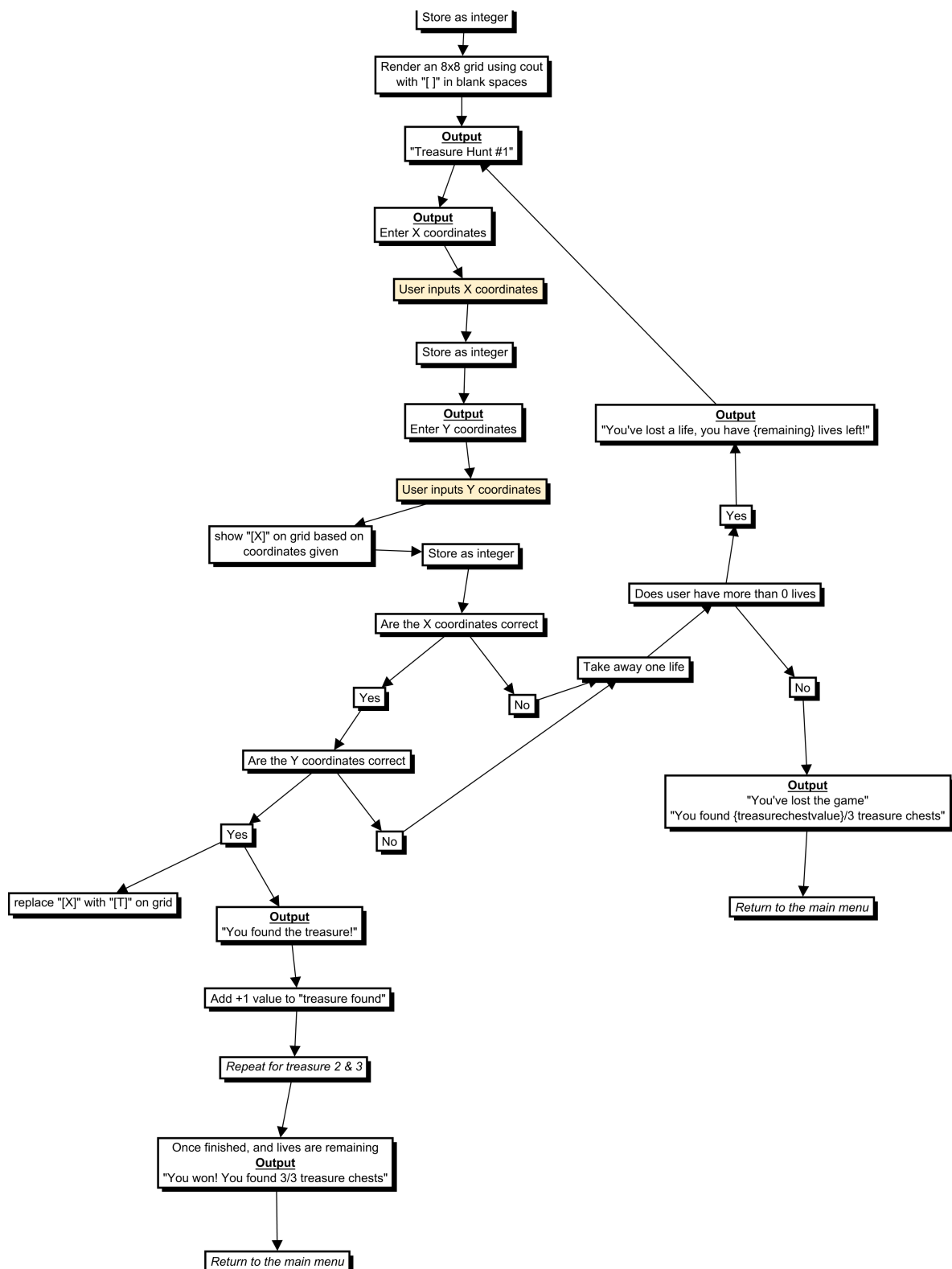
```
1  SetConsoleTitle(TEXT("Exercise 3 - Tomos's Assignment"));
2      system("cls");
3      cout << "Please enter an arbitrary set of prices" << endl;
4      double prices[5];
5
6      int index;
7      for (index = 0; index < 5; index++)
8      {
9          cout << "Enter price number " << (index + 1) << ": ";
10         cin >> prices[index];
11         while (!cin)
12         {
13             cout << "[!] Invalid data type! Please try again.";
14             cin.clear();
15             cin.ignore();
16             cin >> prices[index];
17         }
18     }
19     system("cls");
20
21     for (index = 0; index < 5; index++)
22     {
23         cout << "Price #" << (index + 1) << ": " << char(156) <<
prices[index] << endl;
24     }
25
26     double total{};
27
28     for (index = 0; index < 5; index++)
```

```
29     {
30         total = total + prices[index];
31     }
32
33     cout << "Total: " << char(156) << total << endl;
34
35     cout << "Total in Euro: " << (total * 1.14148) << " euros"
<<endl;
36
37     system("pause");
38     system("cls");
39     mainMenu();
```

Exercise 4 (Attempt)

Preparations





Problems

The developer was uncertain on how to develop this exercise. Issues arose such as not being able to translate the concept map developed as preparation (see above) to actual C++ code.

Problem #1

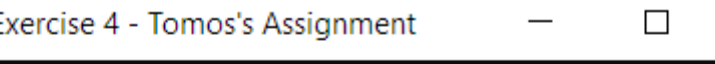
Validating the user input for the program would be counter intuitive, as the developer's methods of validation is to restart the form when an error arises, but, since the form is so big for this particular program, a solution was needed. Luckily, the `goto` command was practiced appropriately.

Problem #2

Entering any data type other than numeric values will loop this error message:

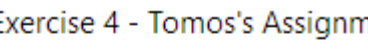
[illegible]

Testing



```
Player 1: Enter your name:  
[!] Player names cannot be blank  
Press any key to continue . . .
```

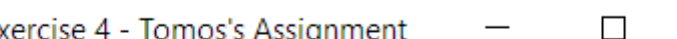
Entering no value at all when asked to enter a player name will force the player to enter a name that isn't blank.



Exercise 4 - Tomos's Assignment

., please look away now.
Press any key to continue

If a player names themselves a single character, the program will still store that value as the player's name

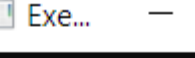


```
Exercise 4 - Tomos's Assignment
Tomas - Treasure #1
Enter X coordinate (0-7): 8
[!] The input must be between 0 and 7
Enter X coordinate (0-7):
```

Entering a value that was out of bounds would throw up an error

[illegible]

Entering a value that isn't a number however will loop an error



```

Treasure #1
Enter X coordinate: 1
Enter Y coordinate: 2
You found the treasure!
Treasure #2
Enter X coordinate: 1

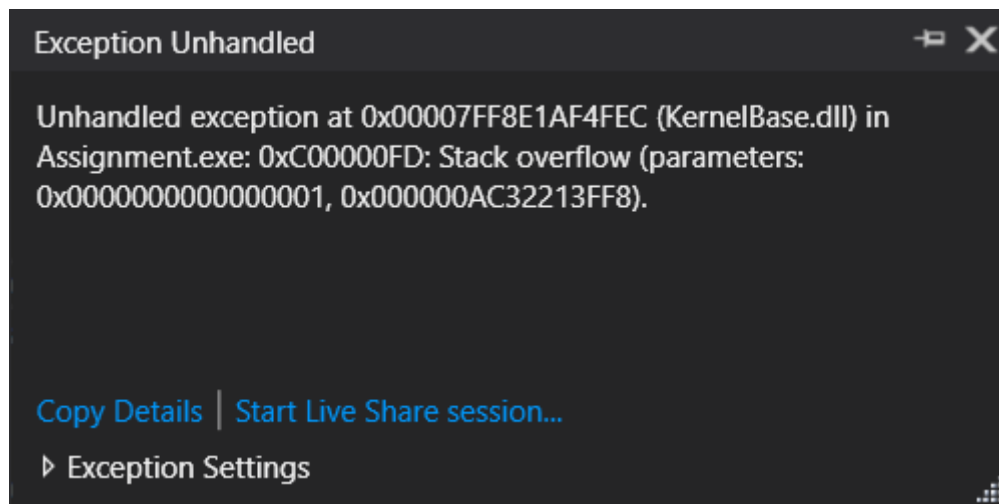
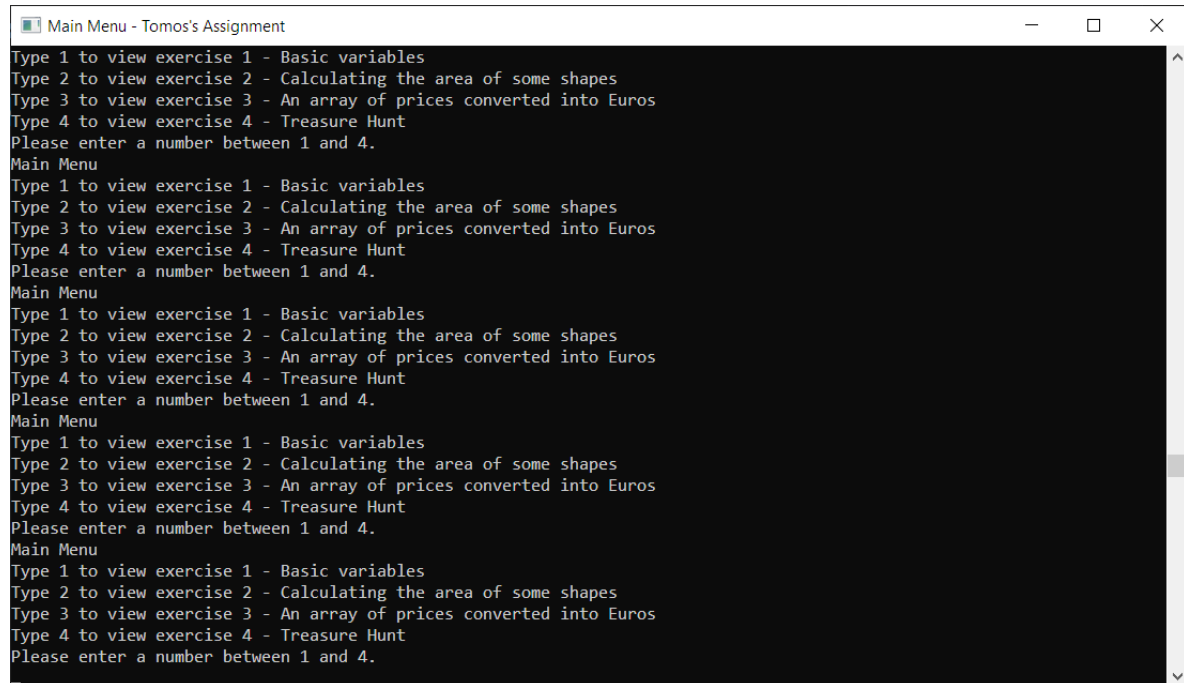
```

Entering the correct value for guessing coordinates functions properly



```
Treasure #1
Enter X coordinate: Enter Y coordinate: There's no treasure here! You lose a life!
You have 1 lives remaining.
Press any key to continue . . .
```

Using invalid data such as characters instead of numbers causes the program to immediately execute 2 lives from the player followed by a massive loop of the main menu that would eventually result in a stack overflow



Code Example

```
1 void exercise4()  
2 {  
3     ///This is only an attempt and therefore does not function  
properly.  
4     SetConsoleTitle(TEXT("Exercise 4 - Tomos's Assignment"));  
5     system("cls");  
6     cout << "Player 1: Enter your name: ";  
7     string playerName;  
8     getline(cin, playerName);  
9     if (playerName.size() < 1)  
10    {  
11        cout << "[!] Player names cannot be blank" << endl;  
12        system("pause");  
13        exercise4();  
14    }  
15    cout << "Player 2: Enter your name: ";
```

```

16     string player2name;
17     getline(cin, player2name);
18     if (player2name.size() < 1)
19     {
20         cout << "[!] Player names cannot be blank" << endl;
21         system("pause");
22         exercise4();
23     }
24     system("cls");
25     cout << player2name << ", " << "please look away now." <<
endl;
26     system("pause");
27     system("cls");
28     int xCoordinateTreasure1;
29     int yCoordinateTreasure1;
30     cout << player1name << " - Treasure #1" << endl;
31     xCoordinateTreasure1:
32     cout << "Enter X coordinate (0-7): ";
33     cin >> xCoordinateTreasure1;
34     if (xCoordinateTreasure1 > 7)
35     {
36         cout << "[!] The input must be between 0 and 7" << endl;
37         goto xCoordinateTreasure1;
38     }
39     if (xCoordinateTreasure1 < 0)
40     {
41         cout << "[!] The input must be between 0 and 7" << endl;
42         goto xCoordinateTreasure1;
43     }
44     yCoordinateTreasure1:
45     cout << "Enter Y coordinate (0-7): ";
46     cin >> yCoordinateTreasure1;
47     if (yCoordinateTreasure1 > 7)
48     {
49         cout << "[!] The input must be between 0 and 7" << endl;
50         goto yCoordinateTreasure1;
51     }
52     if (yCoordinateTreasure1 < 0)
53     {
54         cout << "[!] The input must be between 0 and 7" << endl;
55         goto yCoordinateTreasure1;
56     }
57     system("cls");
58     int xCoordinateTreasure2;
59     int yCoordinateTreasure2;
60     cout << player1name << " - Treasure #2" << endl;
61     xCoordinateTreasure2:
62     cout << "Enter X coordinate (0-7): ";
63     cin >> xCoordinateTreasure2;
64     if (xCoordinateTreasure2 > 7)
65     {
66         cout << "[!] The input must be between 0 and 7" << endl;
67         goto xCoordinateTreasure2;
68     }
69     if (xCoordinateTreasure2 < 0)

```

```

70     {
71         cout << "[!] The input must be between 0 and 7" << endl;
72         goto xCoordinateTreasure2;
73     }
74 yCoordinateTreasure2:
75     cout << "Enter Y coordinate (0-7): ";
76     cin >> yCoordinateTreasure2;
77     if (yCoordinateTreasure2 > 7)
78     {
79         cout << "[!] The input must be between 0 and 7" << endl;
80         goto yCoordinateTreasure2;
81     }
82     if (yCoordinateTreasure2 < 0)
83     {
84         cout << "[!] The input must be between 0 and 7" << endl;
85         goto yCoordinateTreasure2;
86     }
87     system("cls");
88     int xCoordinateTreasure3;
89     int yCoordinateTreasure3;
90     cout << player1name << " - Treasure #3" << endl;
91     xCoordinateTreasure3:
92     cout << "Enter X coordinate (0-7): ";
93     cin >> xCoordinateTreasure3;
94     if (xCoordinateTreasure3 > 7)
95     {
96         cout << "[!] The input must be between 0 and 7" << endl;
97         goto xCoordinateTreasure3;
98     }
99     if (xCoordinateTreasure3 < 0)
100    {
101        cout << "[!] The input must be between 0 and 7" << endl;
102        goto xCoordinateTreasure3;
103    }
104    yCoordinateTreasure3:
105    cout << "Enter Y coordinate (0-7): ";
106    cin >> yCoordinateTreasure3;
107    if (yCoordinateTreasure3 > 7)
108    {
109        cout << "[!] The input must be between 0 and 7" << endl;;
110        goto yCoordinateTreasure3;
111    }
112    if (yCoordinateTreasure3 < 0)
113    {
114        cout << "[!] The input must be between 0 and 7" << endl;
115        goto yCoordinateTreasure3;
116    }
117    system("pause");
118    system("cls");
119    int lives = 3;
120    cout << player2name << " - You can now look at the display."
<< endl;
121    cout << player1name << " has hidden 3 sets of treasure around
this 8x8 map" << endl;

```

```

122     cout << "You have 3 lives, you must find the treasure!" <<
endl;
123     system("pause");
124     int foundtreasure = 0;
125 level1:
126     int guessXCoordinate1;
127     int guessYCoordinate1;
128     system("cls");
129     cout << "Treasure #1" << endl;
130     cout << "Enter X coordinate: ";
131     cin >> guessXCoordinate1;
132     cout << "Enter Y coordinate: ";
133     cin >> guessYCoordinate1;
134
135     if (guessXCoordinate1 == xCoordinateTreasure1)
136     {
137         if (guessYCoordinate1 == yCoordinateTreasure1)
138         {
139             cout << "You found the treasure!" << endl;
140             foundtreasure = foundtreasure + 1;
141             goto level2;
142         }
143     }
144     else
145     {
146         if (lives == 1)
147         {
148             goto gameover;
149         }
150         else
151         {
152             cout << "There's no treasure here! You lose a life!" <<
endl;
153             lives = lives - 1;
154             cout << "You have " << lives << " lives remaining." <<
endl;
155             system("pause");
156             goto level1;
157         }
158     }
159 }
160
161 level2:
162     int guessXCoordinate2;
163     int guessYCoordinate2;
164     cout << "Treasure #2" << endl;
165     cout << "Enter X coordinate: ";
166     cin >> guessXCoordinate2;
167     cout << "Enter Y coordinate: ";
168     cin >> guessYCoordinate2;
169
170     if (guessXCoordinate2 == xCoordinateTreasure2)
171     {
172         if (guessYCoordinate2 == yCoordinateTreasure2)
173         {

```

```

174         cout << "You found the treasure!" << endl;
175         foundtreasure = foundtreasure + 1;
176         goto level3;
177     }
178 }
179 else
180 {
181     if (lives == 1)
182     {
183         goto gameover;
184     }
185     cout << "There's no treasure here! You lose a life!" <<
endl;
186     lives = lives - 1;
187     cout << "You have " << lives << " lives remaining." <<
endl;
188     system("pause");
189     goto level2;
190 }
191
192 level3:
193     int guessXCoordinate3;
194     int guessYCoordinate3;
195     cout << "Treasure #3" << endl;
196     cout << "Enter X coordinate: ";
197     cin >> guessXCoordinate3;
198     cout << "Enter Y coordinate: ";
199     cin >> guessYCoordinate3;
200
201     if (guessXCoordinate3 == xCoordinateTreasure3)
202     {
203         if (guessYCoordinate3 == yCoordinateTreasure3)
204         {
205             cout << "You found the treasure!" << endl;
206             foundtreasure = foundtreasure + 1;
207             goto finale;
208         }
209     }
210     else
211     {
212         if (lives == 1)
213         {
214             goto gameover;
215         }
216         cout << "There's no treasure here! You lose a life!" <<
endl;
217         lives = lives - 1;
218         cout << "You have " << lives << " lives remaining." <<
endl;
219         system("pause");
220         goto level3;
221     }
222
223 finale:

```

```
224     cout << "You found " << foundtreasure << "/3 treasure chests"
    << endl;
225     system("pause");
226     system("cls");
227     mainMenu();
228
229 gameover:
230     cout << "You've lost the game!" << endl;
231     goto finale;
232 }
```