

Worksheet 1 Testing

by Sam Mifsud

Submission date: 13-Jan-2024 07:22PM (UTC+0000)

Submission ID: 221853437

File name: 50540_Sam_Mifsud_Worksheet_1_Testing_541134_653348165.docx (133.14K)

Word count: 658

Character count: 3242

WORKSHEET 1 – TESTING (Using Variables)

Original code:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

void main()
{
    char surname[20];
    char insuranceNumber[13];
    int age = 0;
    int validationLoop = 1;

    while (validationLoop == 1)
    {
        printf("Please enter your surname: ");
        scanf("%s", &surname);
        if (strcmp(surname, " ") == 0)
        {
            printf("Invalid Input! Input cannot be blank. \n");
        }
        else
        {
            break;
        }
    }
    while (validationLoop == 1)
    {
        printf("Please enter your national Insurance Number: ");
        scanf("%s", &insuranceNumber);
        if (strcmp(insuranceNumber, " ") == 0)
        {
            printf("Invalid Input! Input cannot be blank. \n");
        }
        else
        {
            break;
        }
    }
    while (validationLoop == 1)
    {
        printf("Please enter your age: ");
        scanf("%d", &age);
        if (age < 1)
        {
            printf("Invalid Input! Age cannot be less than 1. \n");
        }
        else
        {
            break;
        }
    }
    printf("%s \n %s \n %d \n", surname, insuranceNumber, age);
}
```


Code Testing

For each test below, run your code and enter the values provided. Provide a screenshot of the programme running. Did the programme behave correctly or did it do something unexpected? Fill in a box for each test below.

Surname:

- i. Enter *JONES*
- ii. Leave blank
- iii. Enter *123*

National Insurance Number:

- i. Enter *PE 25 34 12 B*
- ii. Leave blank
- iii. Enter *AB 11 22 33 44 55 66 77* (ie too long for National Insurance number format)

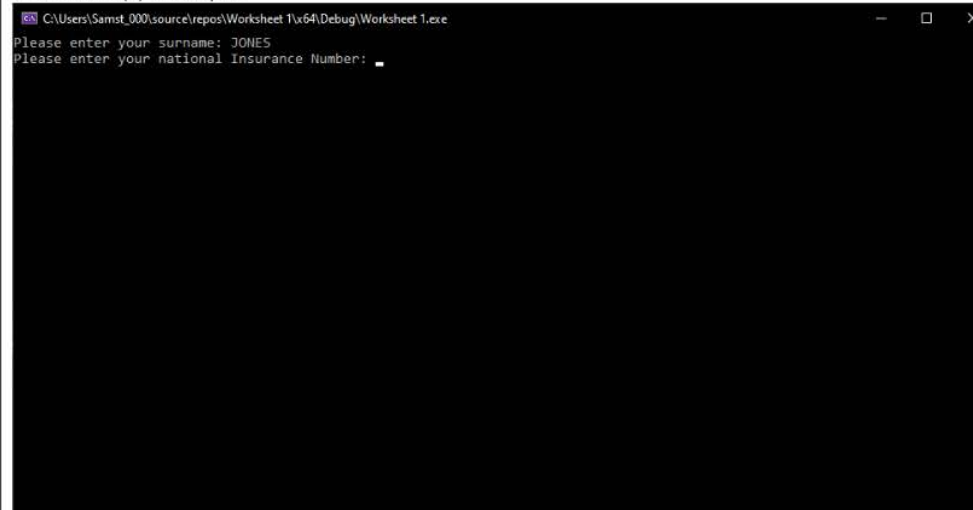
Age

- i. Enter *34*
- ii. Enter *150*
- iii. Enter *-4*
- iv. Enter a letter

Test:

Surname i. Enter *JONES*

Screenshot(s) of output:



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

YES

JONES would be a valid surname, so the program continues.

Test:

Surname ii. Leave blank

Screenshot(s) of output:



```
Select C:\Users\Samst_000\source\repos\Worksheet 1\64\Debug\Worksheet 1.exe
Please enter your surname:
_
```

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

YES

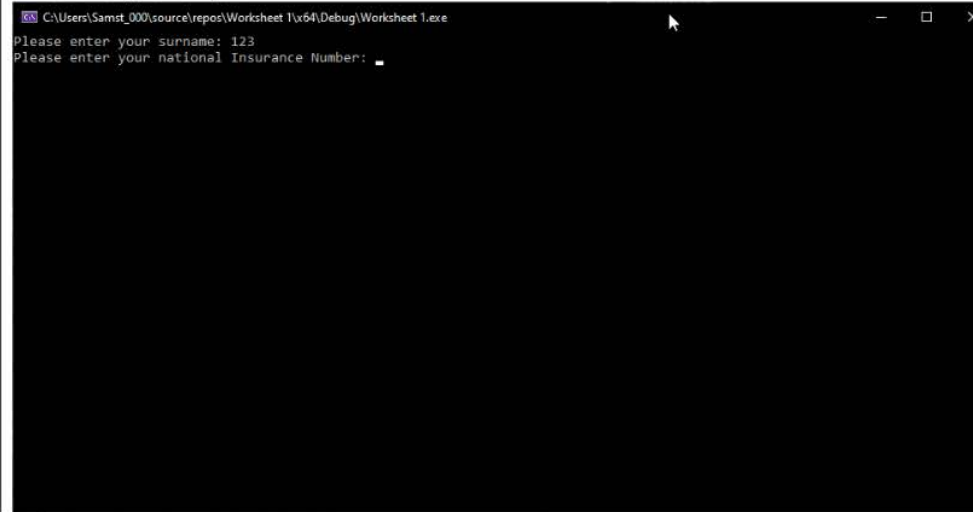
The program doesn't accept the input and puts the cursor on the next line. It will continue to do this until the user enters a valid input.

Continue the rest of the tests below. (Copy and paste a box for each one)

Test:

Surname iii. Enter 123

Screenshot(s) of output:



```
C:\Users\Samst_000\source\repos\Worksheet 1\64\Debug\Worksheet 1.exe
Please enter your surname: 123
Please enter your national Insurance Number:
_
```

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

YES

While it may seem incorrect to allow a user to enter a number as their surname there are people with numbers in their surnames. Preventing these people from entering their actual legal surnames could cause problems, so this input has to be allowed.



Test:

National Insurance Number i. Enter PE 25 34 12 B

Screenshot(s) of output:

```
Microsoft Visual Studio Debug Console
Please enter your surname: 123
Please enter your national Insurance Number: PE 25 34 12 B
Please enter your age: 123
PE
25
C:\Users\Samst_000\source\repos\Worksheet 1\Debug\Worksheet 1.exe (process 9268) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

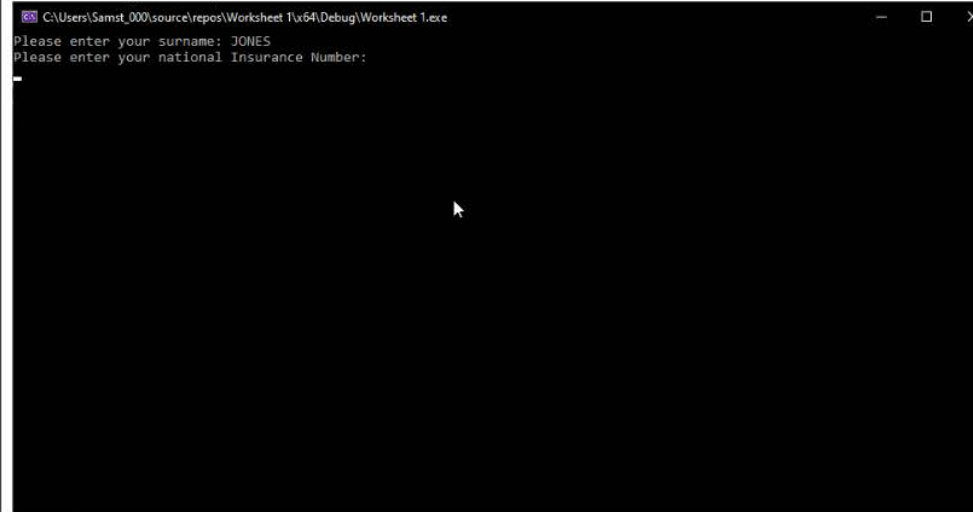
NO

Due to the scanf function handling spaces as newline characters the program breaks, inputting part of the string into the next scanf call and causing problems with the program.

Test:

National Insurance Number ii. Leave blank

Screenshot(s) of output:



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

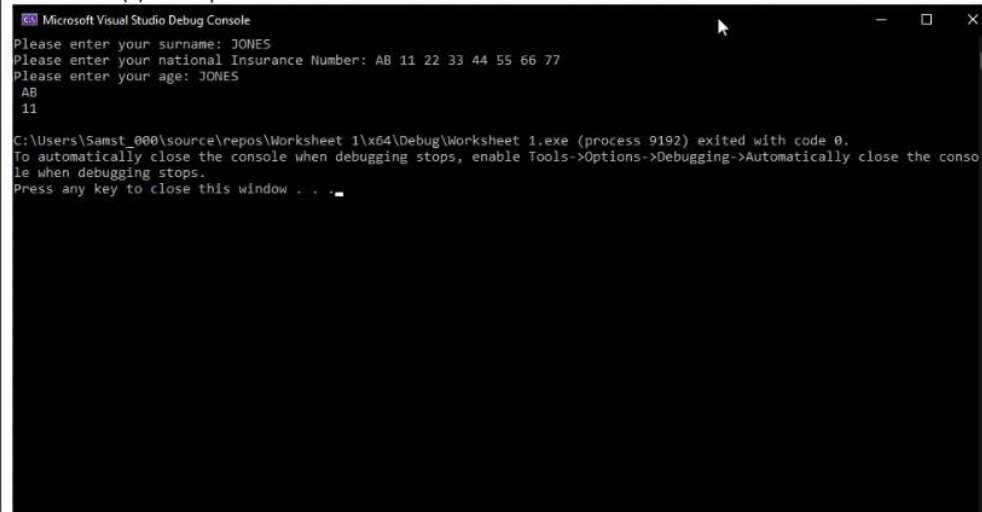
YES

The program doesn't accept the input and puts the cursor on the next line. It will continue to do this until the user enters a valid input.

Test:

National Insurance Number iii. Enter AB 11 22 33 44 55 66 77 (ie too long for National Insurance number format)

Screenshot(s) of output:



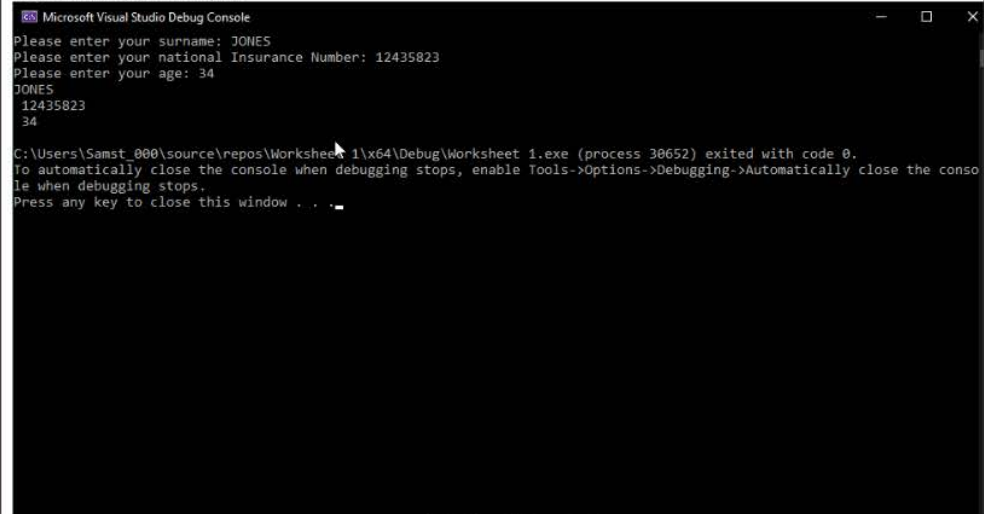
Result: Did the programme behave as expected ie syntactically and logically correct?

NO

Due to the scanf function handling spaces as newline characters the program breaks, inputting part of the string into the next scanf call and causing problems with the program.

Test:
Age i. Enter 34

Screenshot(s) of output:

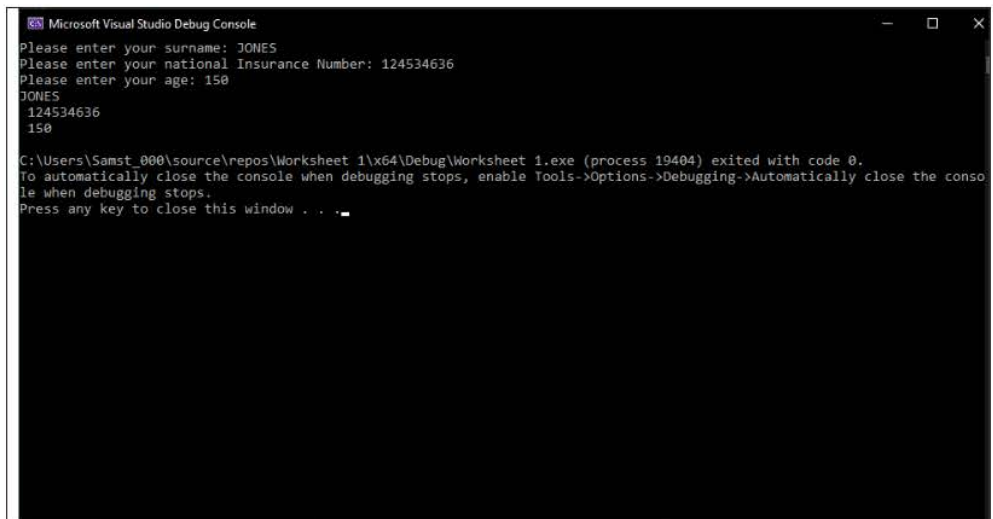


```
Microsoft Visual Studio Debug Console
Please enter your surname: JONES
Please enter your national Insurance Number: 12435823
Please enter your age: 34
JONES
12435823
34
C:\Users\Samst_000\source\repos\Worksheet 1\Debug\Worksheet 1.exe (process 30652) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Result: Did the programme behave as expected ie syntactically and logically correct?
YES
34 is a valid input and the program handles it correctly.

Test:
Age ii. Enter 150

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Please enter your surname: JONES
Please enter your national Insurance Number: 124534636
Please enter your age: 150
JONES
124534636
150
C:\Users\Samst_000\source\repos\Worksheet 1\Debug\Worksheet 1.exe (process 19404) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

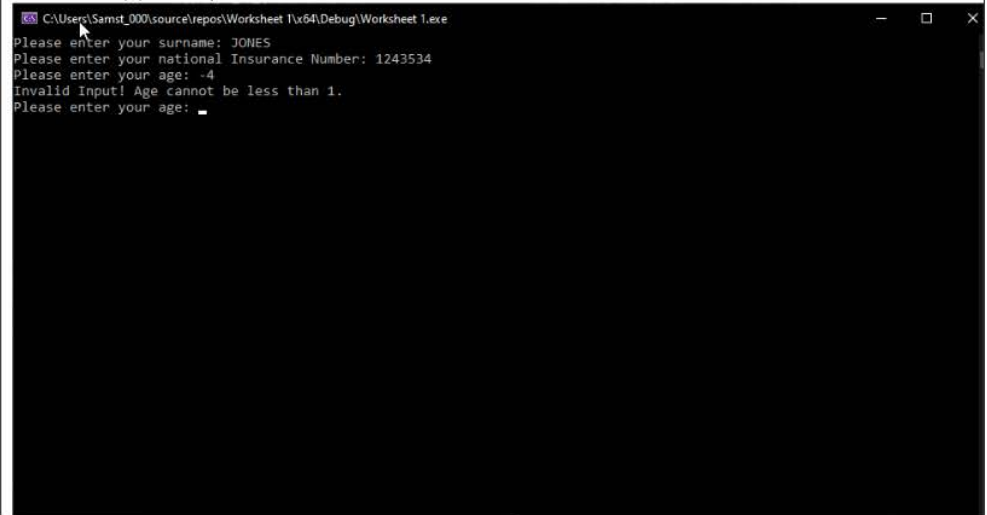
NO

While a person could in theory live to the age of 150 it is unlikely, and the oldest known person in the world right now is only 116 years old (https://en.wikipedia.org/wiki/Oldest_people), so allowing an age this high is illogical.

Test:

Age iii. Enter -4

Screenshot(s) of output:



```
C:\Users\Samst_000\source\repos\Worksheet 1\Debug\Worksheet 1.exe
Please enter your surname: JONES
Please enter your national Insurance Number: 1243534
Please enter your age: -4
Invalid Input! Age cannot be less than 1.
Please enter your age: _
```

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

YES

The program prevents the input telling the user “Age cannot be less than 1” and asks them to enter another input.

Age iii. Enter a letter

[illegible]

NO

9

1. Consider the results of the above tests and revisit your code in order to solve the problems.
(Note: you may not be able to solve all of them, but solve as many as you can.)

Paste your revised code here:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

void main()
{
    char surname[20];
    char insuranceNumber[14];
    int age = 0;
    int validationLoop = 1;

    while (validationLoop == 1)
    {
        printf("Please enter your surname: ");
        scanf("%s", &surname);
        if (strcmp(surname, " ") == 0)
        {
            printf("Invalid Input! Input cannot be blank. \n");
        }
        else
        {
            break;
        }
    }

    while (validationLoop == 1)
    {
        printf("Please enter your National Insurance Number: ");
        scanf("%[^\n]s", &insuranceNumber);
        if (strcmp(insuranceNumber, " ") == 0)
        {
            printf("Invalid Input! Input cannot be blank. \n");
        }
        else
        {
            break;
        }
    }

    while (validationLoop == 1)
    {
        printf("Please enter your age: ");
        scanf("%d", &age);
        if (age < 1)
        {
            printf("Invalid Input! Age cannot be less than 1. \n");
        }
        else if (age > 120)
        {
            printf("Invalid Input! Age cannot be greater than 120. \n");
        }
        else
        {
            break;
        }
    }

    printf("%s\n%s\n%d\n", surname, insuranceNumber, age);
}
```

Worksheet 1 Testing

GRADEMARK REPORT

FINAL GRADE

/10

GENERAL COMMENTS

Worksheet 1:

Coding: Good. I see you have made progress with the non-entry of spaces. If you were going to implement this in a live system, you would need to look at the format of the National Insurance number itself, rather than just whether it is blank or not.

4/5

Testing:

Good, you have understood this section well and suitably presented your results.

5/5

Total: 9/10

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1

Fair comment, but there has to be a balance. Could someone's surname really be 123? Or is it more likely to be a typo?

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

Worksheet 2 Testing

by Sam Mifsud

Submission date: 13-Jan-2024 11:50PM (UTC+0000)

Submission ID: 221864838

File name: 50540_Sam_Mifsud_Worksheet_2_Testing_541134_706606037.docx (136.12K)

Word count: 444

Character count: 2254

WORKSHEET 2 – TESTING (Carrots)

1. Original code:

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

void main()
{
    int requiredSize = 8;
    int bedLength = 0;
    int bedWidth = 0;
    int carrotsWide = 0;
    int carrotsLong = 0;
    int validationLoop = 0;

    while (validationLoop == 0)
    {
        printf("Please enter the length of your vegetable bed to the nearest whole centimeter: ");
        scanf("%d", &bedLength);
        if (bedLength < 1)
        {
            printf("Invalid Input! Bed length cannot be less than 1cm. \n");
        }
        else
        {
            break;
        }
    }

    while (validationLoop == 0)
    {
        printf("Please enter the width of your vegetable bed to the nearest whole centimeter: ");
        scanf("%d", &bedWidth);
        if (bedWidth < 1)
        {
            printf("Invalid Input! Bed width cannot be less than 1cm. \n");
        }
        else
        {
            carrotsLong = bedLength / requiredSize;
            carrotsWide = bedWidth / requiredSize;
            int total = carrotsLong * carrotsWide;
            int carrotLength = carrotsLong * requiredSize;
            int carrotWidth = carrotsWide * requiredSize;
            int lengthLeft = bedLength - carrotLength;
            int widthLeft = bedWidth - carrotWidth;
            int carrotArea = carrotLength * carrotWidth;
            int areaLeft = lengthLeft * widthLeft;
            printf("You can fit %d carrots in your bed. You can fit %d lengthways and %d\n", total, carrotsLong, carrotsWide);
            printf("You will have %dcm left lengthways and %dcm left widthways. The total left\n", lengthLeft, widthLeft, areaLeft);
            printf("The carrots will take up %dcm lengthways and %dcm widthways. They will take\n", carrotLength, carrotWidth, carrotArea);
            break;
        }
    }
}

```

2. Code Testing

For each test below, run your code and enter the values provided. Provide a screenshot of the programme output. Did the programme behave correctly or did it do something unexpected? Fill in a box for each test below.

Input:

- i. Enter dimensions: Width 45 Length 85
- ii. Enter dimensions: Width 7 Length 25
- iii. Enter dimensions: Width -16 Length -20

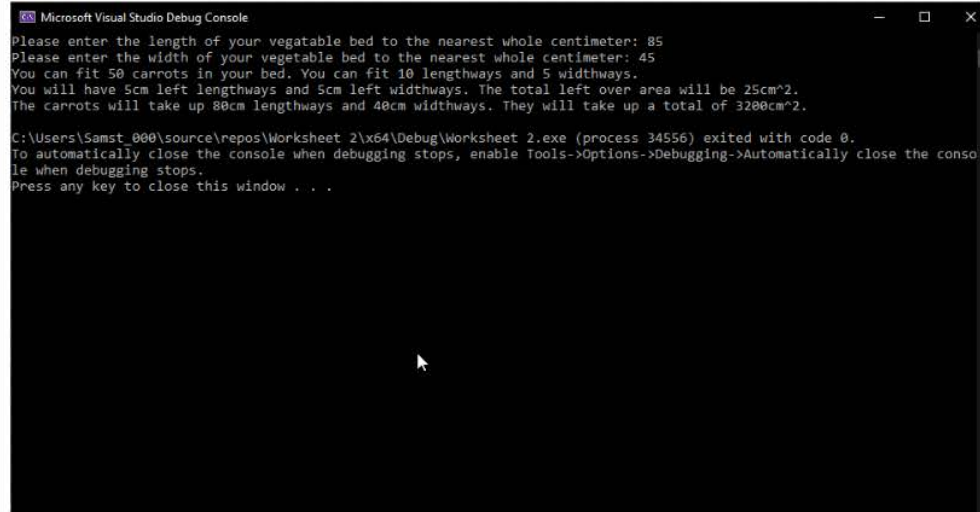
Logic:

- i. Enter dimensions: Width 27 Length 30. The result should be 9 carrots.
- ii. Enter dimensions: Width 32 Length 32. The result should be 16 carrots.
- iii. Enter dimensions: Width 7 Length 16. The result should either be 0 carrots, or you may have accounted for this in your validation at the input phase.

Test:

Input - Enter dimensions: width 45 length 85

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Please enter the length of your vegetable bed to the nearest whole centimeter: 85
Please enter the width of your vegetable bed to the nearest whole centimeter: 45
You can fit 50 carrots in your bed. You can fit 10 lengthways and 5 widthways.
You will have 5cm left lengthways and 5cm left widthways. The total left over area will be 25cm^2.
The carrots will take up 80cm lengthways and 40cm widthways. They will take up a total of 3200cm^2.
C:\Users\Samst.000\source\repos\Worksheet 2\x64\Debug\Worksheet 2.exe (process 34556) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

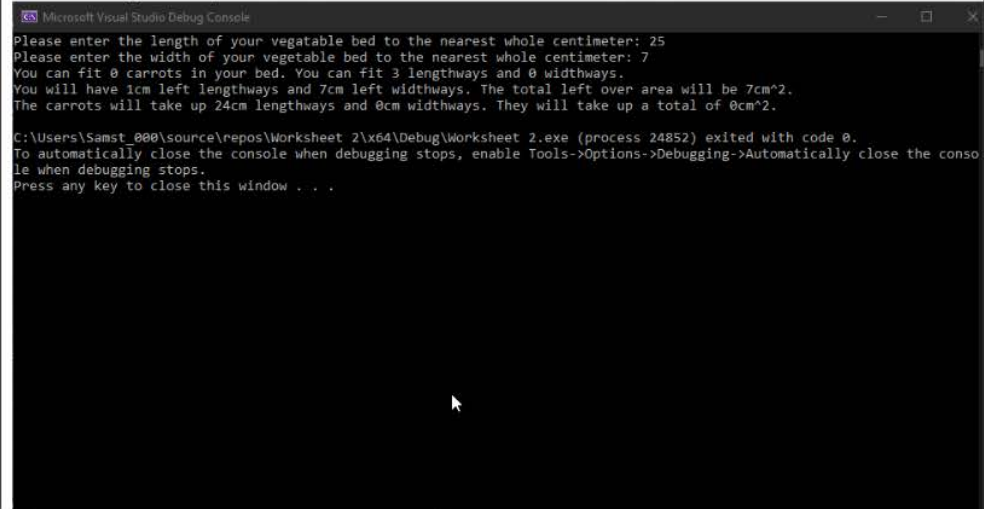
YES

The inputs are accepted and the calculations are performed as expected.

Test:

Enter dimensions: width 7 length 25

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Please enter the length of your vegetable bed to the nearest whole centimeter: 25
Please enter the width of your vegetable bed to the nearest whole centimeter: 7
You can fit 0 carrots in your bed. You can fit 3 lengthways and 0 widthways.
You will have 1cm left lengthways and 7cm left widthways. The total left over area will be 7cm^2.
The carrots will take up 24cm lengthways and 0cm widthways. They will take up a total of 0cm^2.

C:\Users\Sams000\source\repos\Worksheet 2\Debug\Worksheet 2.exe (process 24852) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

YES

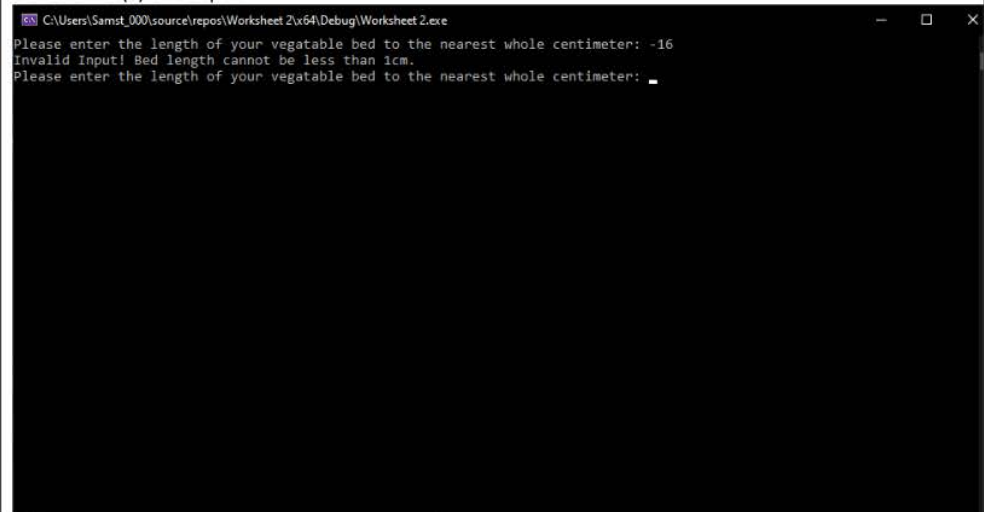
The inputs are valid and the program accepts them and performed the calculations as expected.

Continue the rest of the tests below. (Copy and paste a box for each one)

Test:

Enter dimensions: Width -16 Length -20

Screenshot(s) of output:



```
C:\Users\Sams000\source\repos\Worksheet 2\Debug\Worksheet 2.exe
Please enter the length of your vegetable bed to the nearest whole centimeter: -16
Invalid Input! Bed length cannot be less than 1cm.
Please enter the length of your vegetable bed to the nearest whole centimeter: _
```

```
C:\Users\Samst_000\source\repos\Worksheet 2\x64\Debug\Worksheet 2.exe
Please enter the length of your vegetable bed to the nearest whole centimeter: -16
Invalid Input! Bed length cannot be less than 1cm.
Please enter the length of your vegetable bed to the nearest whole centimeter: 16
Please enter the width of your vegetable bed to the nearest whole centimeter: -20
Invalid Input! Bed width cannot be less than 1cm.
Please enter the width of your vegetable bed to the nearest whole centimeter: _
```

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

YES

The inputs, being invalid, were rejected, and the user prompted to enter a valid input instead. The program would not perform the calculations until valid inputs were entered.

Test:

Enter dimensions: Width 27 Length 30. The result should be 9 carrots

Screenshot(s) of output:

```
Microsoft Visual Studio Debug Console
Please enter the length of your vegetable bed to the nearest whole centimeter: 30
Please enter the width of your vegetable bed to the nearest whole centimeter: 27
You can fit 9 carrots in your bed. You can fit 3 lengthways and 3 widthways.
You will have 6cm left lengthways and 3cm left widthways. The total left over area will be 18cm^2.
The carrots will take up 24cm lengthways and 24cm widthways. They will take up a total of 576cm^2.

C:\Users\Samst_000\source\repos\Worksheet 2\x64\Debug\Worksheet 2.exe (process 11144) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

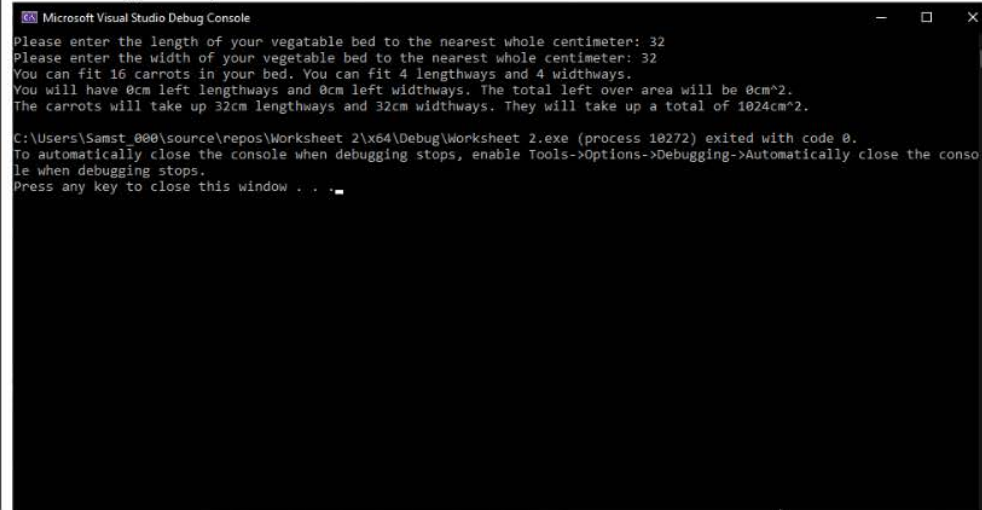
YES

The correct results are calculated and output

Test:

Enter dimensions: Width 32 Length 32. The result should be 16 carrots

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Please enter the length of your vegetable bed to the nearest whole centimeter: 32
Please enter the width of your vegetable bed to the nearest whole centimeter: 32
You can fit 16 carrots in your bed. You can fit 4 lengthways and 4 widthways.
You will have 0cm left lengthways and 0cm left widthways. The total left over area will be 0cm^2.
The carrots will take up 32cm lengthways and 32cm widthways. They will take up a total of 1024cm^2.

C:\Users\Samst_000\source\repos\Worksheet 2\x64\Debug\Worksheet 2.exe (process 10272) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

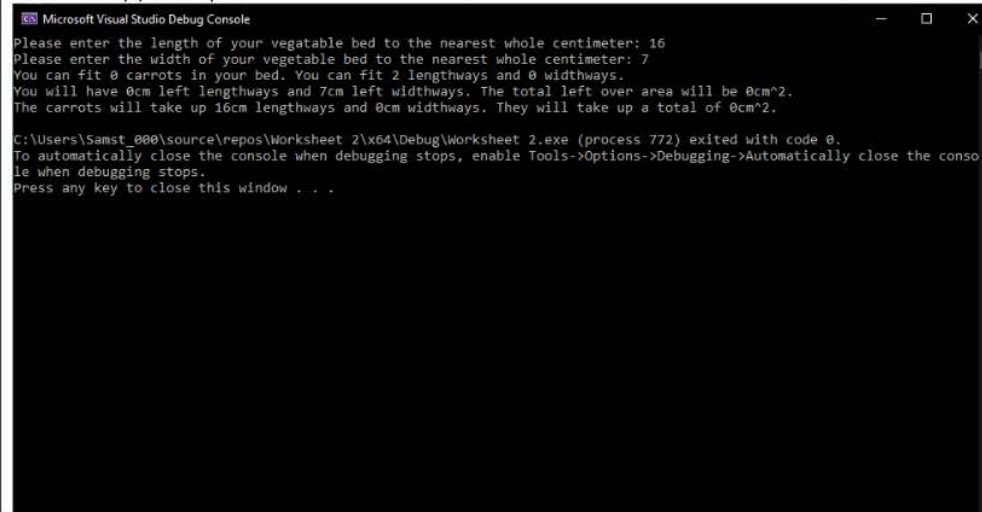
YES

The correct results were calculated and output.

Test:

Enter dimensions: Width 7 Length 16. The result should either be 0 carrots, or you may have accounted for this in your validation at the input phase

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Please enter the length of your vegetable bed to the nearest whole centimeter: 16
Please enter the width of your vegetable bed to the nearest whole centimeter: 7
You can fit 0 carrots in your bed. You can fit 2 lengthways and 0 widthways.
You will have 0cm left lengthways and 7cm left widthways. The total left over area will be 0cm^2.
The carrots will take up 16cm lengthways and 0cm widthways. They will take up a total of 0cm^2.

C:\Users\Samst_000\source\repos\Worksheet 2\x64\Debug\Worksheet 2.exe (process 772) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

YES

The program correctly outputs that no carrots can be fit, as there is not enough space widthways.

3. Consider the results of the above tests and revisit your code in order to solve any problems discovered at the testing phase. (Note: you may not be able to solve all of them, but solve as many as you can.)

Paste your revised code here:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

void main()
{
    int requiredSize = 8;
    int bedLength = 0;
    int bedWidth = 0;
    int carrotsWide = 0;
    int carrotsLong = 0;
    int validationLoop = 0;

    while (validationLoop == 0)
    {
        printf("Please enter the length of your vegetable bed to the nearest whole centimeter: ");
        scanf("%d", &bedLength);
        if (bedLength < 1)
        {
            printf("Invalid Input! Bed length cannot be less than 1cm. \n");
        }
        else
        {
            break;
        }
    }

    while (validationLoop == 0)
    {
        printf("Please enter the width of your vegetable bed to the nearest whole centimeter: ");
        scanf("%d", &bedWidth);
        if (bedWidth < 1)
        {
            printf("Invalid Input! Bed width cannot be less than 1cm. \n");
        }
        else
        {
            carrotsLong = bedLength / requiredSize;
            carrotsWide = bedWidth / requiredSize;
            int total = carrotsLong * carrotsWide;
            int carrotLength = carrotsLong * requiredSize;
            int carrotWidth = carrotsWide * requiredSize;
            int lengthLeft = bedLength - carrotLength;
            int widthLeft = bedWidth - carrotWidth;
            int carrotArea = carrotLength * carrotWidth;
            int areaLeft = lengthLeft * widthLeft;
            if (total > 0)
            {
                printf("You can fit %d carrots in your bed. You can fit %d lengthways and %d widthways. \n", total,
                    carrotsLong, carrotsWide);
                printf("You will have %dcm left lengthways and %dcm left widthways. The total left over area will
                    be %dcm^2. \n", lengthLeft, widthLeft, areaLeft);
                printf("The carrots will take up %dcm lengthways and %dcm widthways. They will take up a total
                    of %dcm^2. \n", carrotLength, carrotWidth, carrotArea);
                break;
            }
            else
            {
                printf("You cannot fit any carrots in this bed as it is too small ");
                if (carrotsLong < 1)
                {
                    printf("lengthways. ");
                }
                else
                {
                    printf("widthways. ");
                }
                break;
            }
        }
    }
}
```

Worksheet 2 Testing

GRADEMARK REPORT

FINAL GRADE

/10

GENERAL COMMENTS

Worksheet 2:

Coding:

The width and length validation should actually be less than 8 - which is the minimum, not 0. You'd solved most of it in class already, well done.

4.5/5

Testing:

Good - shows original code was quite robust in itself.

5/5

Total: 9.5/10

PAGE 1

PAGE 2



Comment 1

Actually, cannot be less than 8cm.



Comment 2

8

PAGE 3

PAGE 4

Worksheet 3 Testing

by Sam Mifsud

Submission date: 14-Jan-2024 12:51AM (UTC+0000)

Submission ID: 221866813

File name: 50540_Sam_Mifsud_Worksheet_3_Testing_541134_203256403.docx (86.63K)

Word count: 438

Character count: 2094

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <math.h>

void main()
{
    char productArray[5][15];
    float priceArray[5];
    float totalPrice = 0;
    float discountPrice = 0;
    int found;
    int count;
    char discount = 'N';
    int validationLoop = 0;

    for (int i = 0; i < 5; i++)
    {
        printf("Please enter the name of item number %d: ", i + 1);
        scanf("%s", &productArray[i]);
        found = 0;
        count = 0;

        while (count < i)
        {
            if (strcmp(productArray[count], productArray[i]) == 0) //Checks if the item has already had a price entered
            {
                priceArray[i] = priceArray[count];
                found = 1;
                break;
            }
            else
            {
                count = count + 1;
            }
        }
        if (found == 0) //if the item has not had a price entered
        {
            while (validationLoop == 0)
            {
                printf("Please enter the price of '%s': ", productArray[i]);
                scanf("%f", &priceArray[i]);
                if (priceArray[i] > 0)
                {
                    break;
                }
                else
                {
                    printf("Invalid Input! Price cannot be 0 or less.\n");
                }
            }
        }
    }

    printf("Does the customer get an employee discount? (Y/N): ");
    scanf("%c", &discount);

    for (int i = 0; i < 5; i++)
    {
        totalPrice = totalPrice + priceArray[i];
    }
    if (discount == 'Y')
    {
        discountPrice = totalPrice - (totalPrice * 0.15);
    }
    else
    {
        discountPrice = totalPrice;
    }

    //Receipt
    printf("RECEIPT\n");
    for (int i = 0; i < 5; i++)
    {
        printf("%s: \x9C%.2f\n", productArray[i], priceArray[i]);
    }
    printf("Total cost: \x9C%.2f\n", discountPrice);
    if (discount == 'Y')
    {
        float discountAmount = totalPrice - discountPrice;
        printf("Employee discount applied! You saved \x9C%.2f", discountAmount);
    }
    else
    {
        printf("No discount applied.");
    }
}

```

1. Original code:

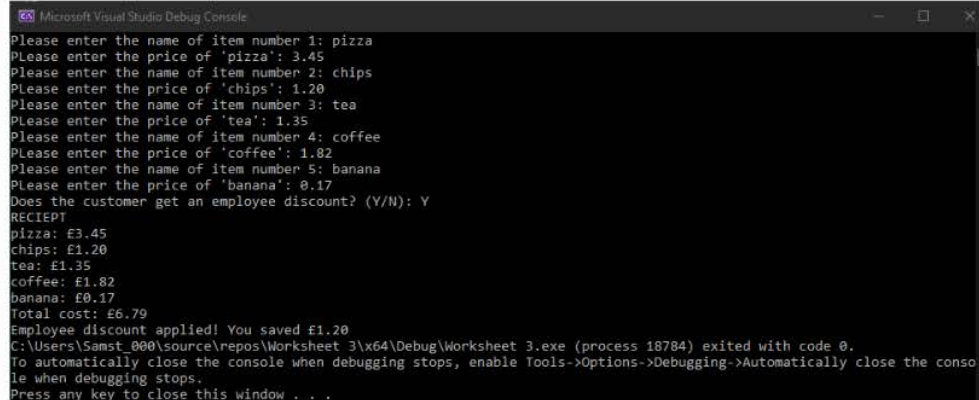
2. Code Testing

For each test below, run your code and enter the values provided. Provide a screenshot of the programme output. Did the programme behave correctly or did it do something unexpected? Fill in a box for each test below.

- i. Enter the 5 normal values as shown on the example screenshot (on the question sheet) ie *pizza 3.45, chips 1.20, tea 1.35, coffee 1.82 and banana 0.17*
Double check the staff discount for the above values, is it correct?
- ii. Enter *pizza -2.34 chips -1.10, tea -1.22, coffee -2.05 and banana -0.18*
- iii. Enter your own values, but leave some of them blank

Test: i. Enter the 5 normal values as shown on the example
ie *pizza 3.45, chips 1.20, tea 1.35, coffee 1.82 and banana 0.17*
Double check the staff discount for the above values, is it correct?

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Please enter the name of item number 1: pizza
Please enter the price of 'pizza': 3.45
Please enter the name of item number 2: chips
Please enter the price of 'chips': 1.20
Please enter the name of item number 3: tea
Please enter the price of 'tea': 1.35
Please enter the name of item number 4: coffee
Please enter the price of 'coffee': 1.82
Please enter the name of item number 5: banana
Please enter the price of 'banana': 0.17
Does the customer get an employee discount? (Y/N): Y
RECEIPT
pizza: £3.45
chips: £1.20
tea: £1.35
coffee: £1.82
banana: £0.17
Total cost: £6.79
Employee discount applied! You saved £1.20
C:\Users\Sams\000\source\repos\Worksheet 3\Debug\Worksheet 3.exe (process 18784) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

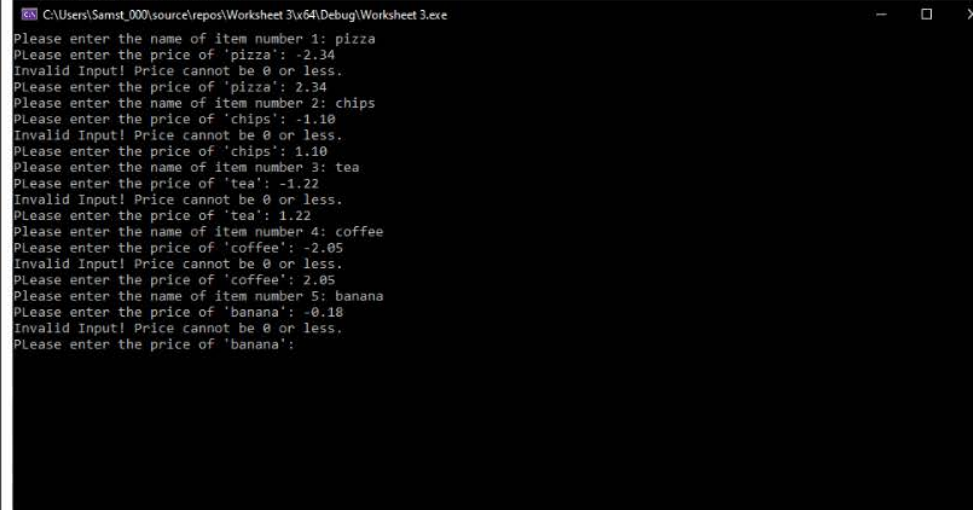
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

The cost of all the items together before the discount is £7.99. The discount is 15% of £7.99, or £1.1985. When rounded this comes out to £1.20. However, the formatting is confusing – the total cost is output after the prices of the individual items (not adjusted for the discount) but is itself adjusted for the discount, which is not specified. This should be changed to make it clearer that this price is the final price including the 15% discount.

Test: ii. Enter *pizza -2.34 chips -1.10, tea -1.22, coffee -2.05 and banana -0.18*

Screenshot(s) of output:



```
C:\Users\Samst_000\source\repos\Worksheet 3\64\Debug\Worksheet 3.exe
Please enter the name of item number 1: pizza
Please enter the price of 'pizza': -2.34
Invalid Input! Price cannot be 0 or less.
Please enter the price of 'pizza': 2.34
Please enter the name of item number 2: chips
Please enter the price of 'chips': -1.10
Invalid Input! Price cannot be 0 or less.
Please enter the price of 'chips': 1.10
Please enter the name of item number 3: tea
Please enter the price of 'tea': -1.22
Invalid Input! Price cannot be 0 or less.
Please enter the price of 'tea': 1.22
Please enter the name of item number 4: coffee
Please enter the price of 'coffee': -2.05
Invalid Input! Price cannot be 0 or less.
Please enter the price of 'coffee': 2.05
Please enter the name of item number 5: banana
Please enter the price of 'banana': -0.18
Invalid Input! Price cannot be 0 or less.
Please enter the price of 'banana':
```

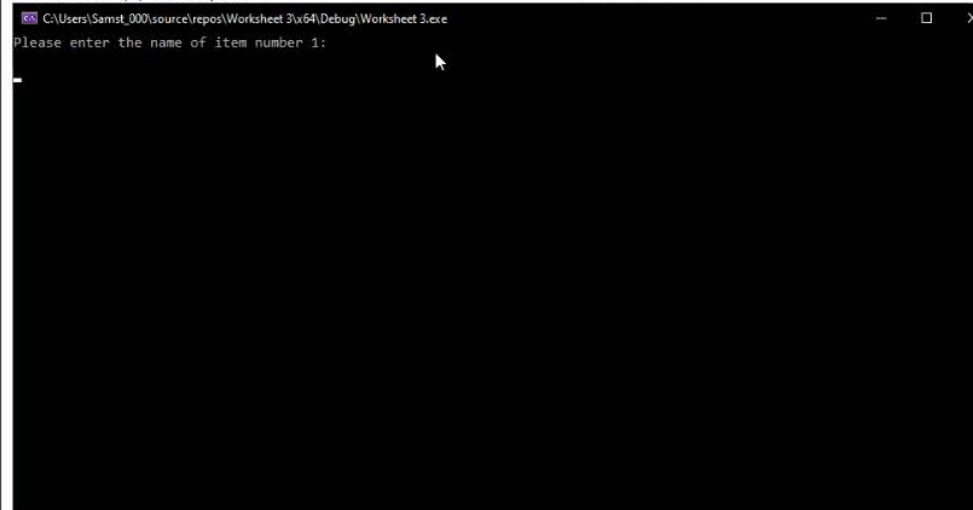
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

Each input does not accept a negative value and instructs the user to input a valid value.

Test: iii) Enter a few of your own values, leave some of the inputs blank

Screenshot(s) of output:



```
C:\Users\Samst_000\source\repos\Worksheet 3\64\Debug\Worksheet 3.exe
Please enter the name of item number 1:

```

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

YES

Inputs cannot be left blank, the cursor will move to the next line but the program will not proceed until a valid input is entered.

Any other tests you feel may be relevant, or reveal a strength or deficiency in your original code, please detail below:

Test:
Screenshot(s) of output:
Result: Did the programme behave as expected ie syntactically and logically correct? YES/NO (delete as appropriate) If NO, what went wrong?

3. Consider the results of the above tests and revisit your code in order to solve any problems discovered at the testing phase. (Note: you may not be able to solve all of them, but solve as many as you can.)

Paste your revised code here:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <math.h>

void main()
{
    char productArray[5][15];
    float priceArray[5];
    float totalPrice = 0;
    float discountPrice = 0;
    int found;
    int count;
    char discount = 'N';
    int validationLoop = 0;

    for (int i = 0; i < 5; i++)
    {
        printf("Please enter the name of item number %d: ", i + 1);
        scanf("%s", &productArray[i]);
        found = 0;
        count = 0;

        while (count < i)
        {
            if (strcmp(productArray[count], productArray[i]) == 0) //Checks if the item has already had a price entered
            {
                priceArray[i] = priceArray[count];
                found = 1;
                break;
            }
            else
            {
                count = count + 1;
            }
        }

        if (found == 0) //if the item has not had a price entered
        {
            while (validationLoop == 0)
            {
                printf("Please enter the price of %s: ", productArray[i]);
                scanf("%f", &priceArray[i]);
                if (priceArray[i] > 0)
                {
                    break;
                }
                else
                {
                    printf("Invalid Input! Price cannot be 0 or less.\n");
                }
            }
        }
    }

    printf("Does the customer get an employee discount? (Y/N): ");
    scanf("%c", &discount);

    for (int i = 0; i < 5; i++)
    {
        totalPrice = totalPrice + priceArray[i];
    }

    //Receipt
    printf("RECEIPT\n");
    for (int i = 0; i < 5; i++)
    {
        printf("%s: \x9C%.2f\n", productArray[i], priceArray[i]);
    }
    printf("Total cost: \x9C%.2f\n", totalPrice);
    if (discount == 'Y')
    {
        discountPrice = totalPrice - (totalPrice * 0.15);
        float discountAmount = totalPrice - discountPrice;
        printf("Employee discount applied! You saved \x9C%.2f\nPrice after discount: \x9C%.2f", discountAmount, discountPrice);
    }
    else
    {
        printf("No discount applied.");
    }
}
```

Worksheet 3 Testing

GRADEMARK REPORT

FINAL GRADE

/20

GENERAL COMMENTS

Worksheet 3:

Coding:

Good. You could also look at not entering a number for a product, but otherwise a complete program, well done.

9/10

Testing:

Presented well, see my comment above.

9/10

Total: 18/20

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

Worksheet 4 Testing

by Sam Mifsud

Submission date: 14-Jan-2024 01:15AM (UTC+0000)

Submission ID: 221867404

File name: 50540_Sam_Mifsud_Worksheet_4_Testing_541134_206818354.docx (107.67K)

Word count: 468

Character count: 2169

WORKSHEET 4a – TESTING (21)

1. Original code:

```
Source.cpp:

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include "header.h"

void main()
{
    srand(time(NULL));
    int tempCard = 0;
    int totalScore = 0;
    int loop = 0;
    char card = 'N';
    int dealerScore;

    printf("Welcome! Welcome to 21! \n\n this game, you are given 3 cards. \nIf you don't bust (in which case you lose) \nor get 21 exactly (in which case you win!) \nyou get to pick whether or not you want more cards\n");
    printf("Aces are low (1)\n");
    printf("Here are your first 3 cards\n");
    for (int i = 0; i < 3; i++)
    {
        tempCard = randInt(10, 1);
        totalScore = totalScore + tempCard;

        printf("You got %d. \n", tempCard);
    }

    while (loop == 0)
    {
        if (totalScore > 21)
        {
            printf("You are bust! You lose :(\n");
            break;
        }
        else if (totalScore == 21)
        {
            printf("You got blackjack. You win 9,837,070 dollar!\n");
            break;
        }
        else
        {
            printf("Your total score is %d. Would you like another card? (Y/N): ", totalScore);
            scanf(" %c", &card);
            if (card == 'N')
            {
                dealerScore = randInt(30, 3);
                printf("Your final score is %d. The dealer got %d. ", totalScore, dealerScore);
                if (((totalScore < 21) and (totalScore > dealerScore)) or dealerScore > 21)
                {
                    printf("You win 4,836 dollar!\n");
                }
                else
                {
                    printf("You lose :(\n");
                }
                break;
            }
            else
            {
                tempCard = randInt(10, 1);
                totalScore = totalScore + tempCard;
                printf("You got %d. \n");
            }
        }
    }
}

header.h:

#pragma once

int randInt(int high, int low)
{
    return rand() % (high - low + 1) + low;
}
```

2. Code Testing

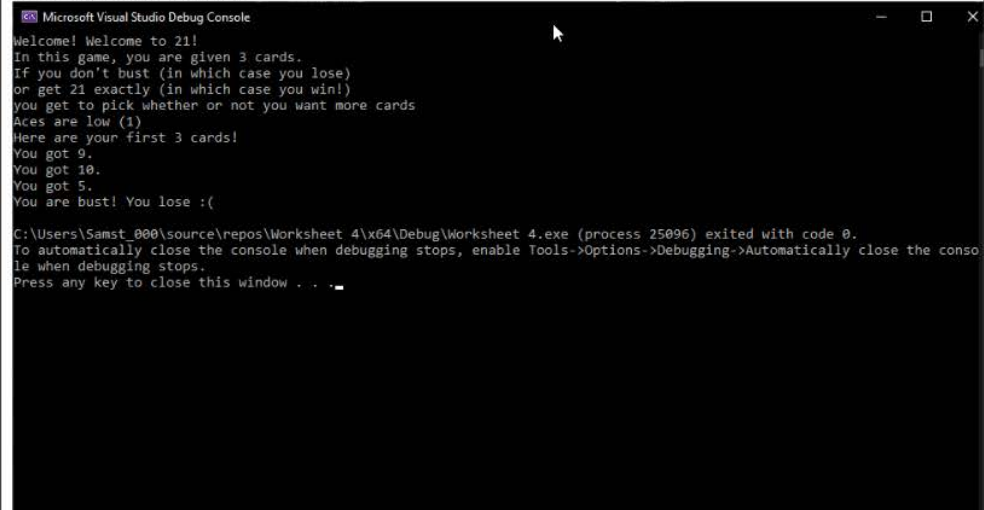
For each test below, run your code and enter the values provided. Provide a screenshot of the programme output. Did the programme behave correctly or did it do something unexpected? Fill in a box for each test below.

Test the following values for your fruit machine:

- i. Bust on first 3 cards drawn (ie total is over 21)
- ii. Bust after user has drawn one or more cards
- iii. User sticks after first 3 cards
- iv. User sticks after they have drawn one or more cards

Test: i. Bust on first 3 cards drawn (ie total is over 21)

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Welcome! Welcome to 21!
In this game, you are given 3 cards.
If you don't bust (in which case you lose)
or get 21 exactly (in which case you win!)
you get to pick whether or not you want more cards
Aces are low (1)
Here are your first 3 cards!
You got 9.
You got 10.
You got 5.
You are bust! You lose :(

C:\Users\Sams000\source\repos\Worksheet 4\Debug\Worksheet 4.exe (process 25096) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

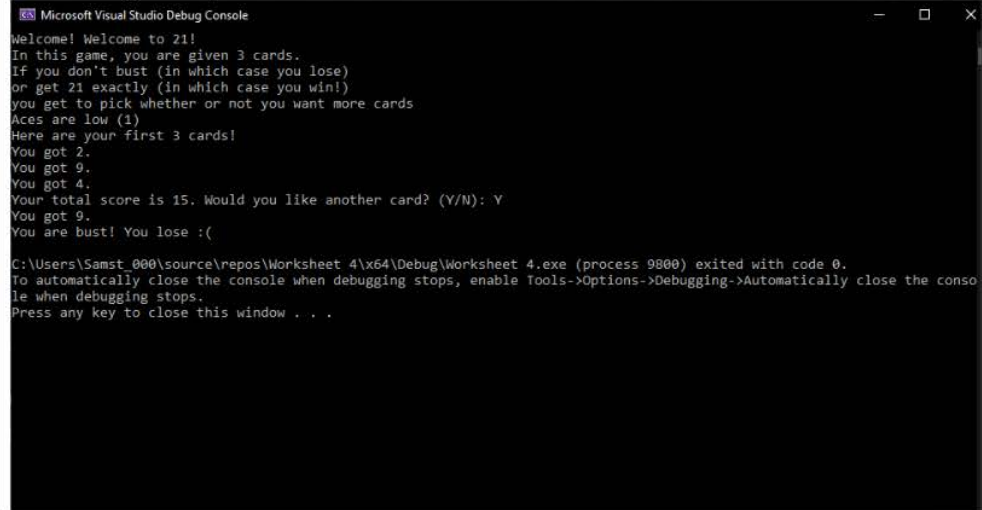
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

The program tells the user they have gone bust and the program exits.

Test: ii. Bust after user has drawn one or more cards

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Welcome! Welcome to 21!
In this game, you are given 3 cards.
If you don't bust (in which case you lose)
or get 21 exactly (in which case you win!)
you get to pick whether or not you want more cards
Aces are low (1)
Here are your first 3 cards!
You got 2.
You got 9.
You got 4.
Your total score is 15. Would you like another card? (Y/N): Y
You got 9.
You are bust! You lose :(

C:\Users\Samst_000\source\repos\Worksheet 4\Debug\Worksheet 4.exe (process 9800) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

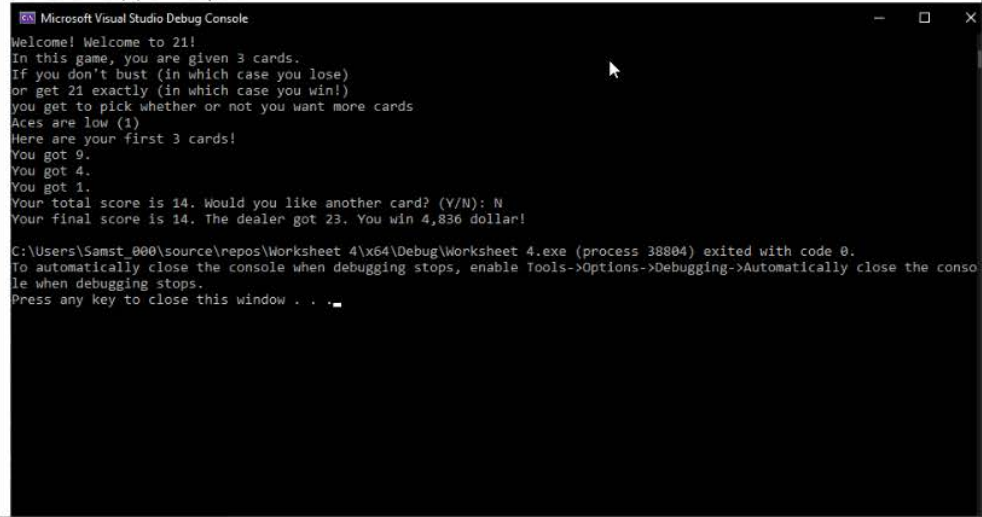
YES

The program tells the user they have gone bust and the program exits.

Continue the rest of the tests below. (Copy and paste a box for each one)

Test: iii. User sticks after first 3 cards

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Welcome! Welcome to 21!
In this game, you are given 3 cards.
If you don't bust (in which case you lose)
or get 21 exactly (in which case you win!)
you get to pick whether or not you want more cards
Aces are low (1)
Here are your first 3 cards!
You got 9.
You got 4.
You got 1.
Your total score is 14. Would you like another card? (Y/N): N
Your final score is 14. The dealer got 23. You win 4,836 dollar!

C:\Users\Samst_000\source\repos\Worksheet 4\Debug\Worksheet 4.exe (process 38804) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

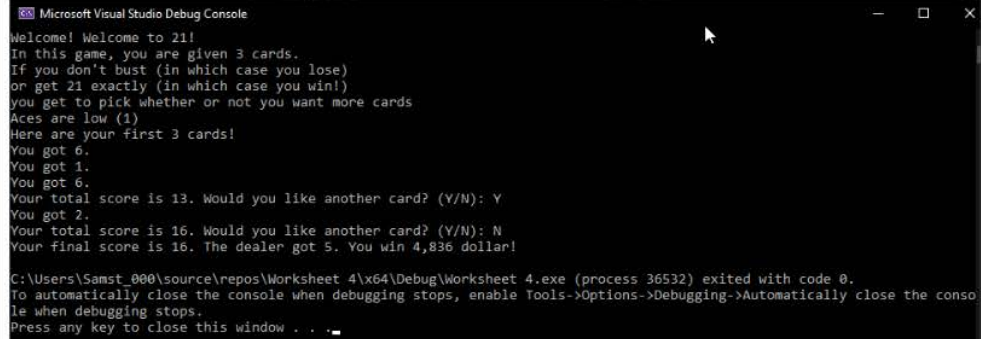
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

The program does not give the user another card. The program then randomly selects a score for the dealer. In this case, the dealer's score was higher than the user's score but was over 21. As the dealer was bust, the user wins a normal prize.

Test: iv. User sticks after they have drawn one or more cards

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console
Welcome! Welcome to 21!
In this game, you are given 3 cards.
If you don't bust (in which case you lose)
or get 21 exactly (in which case you win!)
you get to pick whether or not you want more cards
Aces are low (1)
Here are your first 3 cards!
You got 6.
You got 1.
You got 6.
Your total score is 13. Would you like another card? (Y/N): Y
You got 2.
Your total score is 16. Would you like another card? (Y/N): N
Your final score is 16. The dealer got 5. You win 4,836 dollar!

C:\Users\Samst_000\source\repos\Worksheet 4\Debug\Worksheet 4.exe (process 36532) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

NO

While the program's logic does calculate correctly, the number output to the user when they draw a card is incorrect. This is because I did not tell the printf statement which value to print. It happens that because the card value was the first one declared it does print it, but the printed value is 1 less than it should be due to the first number in code being 0. This can easily be fixed by simply telling the printf statement which value to print.

Any other tests you feel may be relevant, or reveal a strength or deficiency in your original code, please detail below:

Test:
Screenshot(s) of output:
Result: Did the programme behave as expected ie syntactically and logically correct? YES/NO (delete as appropriate) If NO, what went wrong?

3. Consider the results of the above tests and revisit your code in order to solve any problems discovered at the testing phase. (Note: you may not be able to solve all of them, but solve as many as you can.)

Paste your revised code here:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include "header.h"

void main()
{
    srand(time(NULL));
    int tempCard = 0;
    int totalScore = 0;
    int loop = 0;
    char card = 'N';
    int dealerScore;

    printf("Welcome! Welcome to 21!\nIn this game, you are given 3 cards. \nIf you don't bust (in which case you lose) \nor get 21 exactly (in which case you win!)
\nyou get to pick whether or not you want more cards!\n");
    printf("Aces are low (1)\n");
    printf("Here are your first 3 cards!\n");
    for (int i = 0; i < 3; i++)
    {
        tempCard = randInt(10, 1);
        totalScore = totalScore + tempCard;

        printf("You got %d. \n", tempCard);
    }

    while (loop == 0)
    {
        if (totalScore > 21)
        {
            printf("You are bust! You lose :( \n");
            break;
        }
        else if (totalScore == 21)
        {
            printf("You got blackjack. You win 9,837,070 dollar!\n");
            break;
        }
        else
        {
            printf("Your total score is %d. Would you like another card? (Y/N): ", totalScore);
            scanf(" %c", &card);
            if (card == 'N')
            {
                dealerScore = randInt(30, 3);
                printf("Your final score is %d. The dealer got %d. ", totalScore, dealerScore);
                if (((totalScore < 21) and (totalScore > dealerScore)) or dealerScore > 21)
                {
                    printf("You win 4,836 dollar!\n");
                }
                else
                {
                    printf("You lose :( \n");
                    break;
                }
            }
            else
            {
                tempCard = randInt(10, 1);
                totalScore = totalScore + tempCard;
                printf("You got %d. \n", tempCard);
            }
        }
    }
}

#pragma once

int randInt(int high, int low)
{
    return rand() % (high - low + 1) + low;
}
```

Worksheet 4 Testing

GRADEMARK REPORT

FINAL GRADE

/20

GENERAL COMMENTS

Worksheet 4:

Coding:

I am not mad about break; it's a little overused here. Nevertheless, good use of the function and you've made it more complex by including the dealer. You might want to print a message for those entering other than "y" or "n"

9/10

Testing:

Good - see point above about "y"/"n" above

9/10

Total: 18/20

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

Worksheet 5 Testing

by Sam Mifsud

Submission date: 15-Jan-2024 02:33AM (UTC+0000)

Submission ID: 221983270

File name: 50540_Sam_Mifsud_Worksheet_5_Testing_541134_1121777375.docx (317.2K)

Word count: 502

Character count: 2580

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <conio.h>

char treasureArray[12][12];
char displayArray[12][12];

#include "Header.h"

void main()
{
    srand(time(NULL));
    int running = 1;
    int validationLoop = 1;
    int tempX = 0;
    int tempY = 0;
    int temp = 0;
    int treasureFound = 1;
    int turns = 0;
    int difficulty = 0;

    printf("Welcome to the treasure hunt V1.1.\n");
    printf("The game will randomly generate a grid. Treasure will be placed randomly on the grid.\n");
    printf("Your task is to find the treasure. Enter the X and Y coordinates you want to check.\n");
    printf("But be careful! There are bombs on the grid. Hitting the bomb means you lose! Good luck!\n");
    printf("\n");
    printf("Key: \n0 = Unchecked \n1 = Checked, Empty \nX = Checked, treasure \n@ = Bomb (You don't want to see this!\n");
    printf("\n");

    printf("Select your difficulty! \n1 = Easy(4x4, 2 treasure 1 bomb) \n2 = Medium(8x8, 4 treasure 4 bombs) \n3 = Hard(12x12, 6 treasure 9 bombs)\n");
    while (validationLoop == 1)
    {
        printf("Your choice: ");
        scanf("%d", &difficulty);
        if (difficulty > 0 and difficulty < 4)
        {
            break;
        }
        else
        {
            printf("Invalid Input! Difficulty must be between 1 and 3.\n");
        }
    }

    //Sets the arrays to blank spaces
    while (tempY < difficulty * 4)
    {
        while (tempX < difficulty * 4)
        {
            displayArray[tempX][tempY] = '0';
            treasureArray[tempX][tempY] = '0';
            tempX++;
        }
        tempX = 0;
        tempY++;
    }
    tempY = 0;

    //Populating the treasure array with treasure
    while (temp < difficulty * 2)
    {
        tempX = randInt(difficulty * 4) - 1, 0;
        tempY = randInt(difficulty * 4) - 1, 0;
        if (treasureArray[tempX][tempY] != 'X')
        {
            treasureArray[tempX][tempY] = 'X';
            temp++;
        }
    }
    tempX = 0;
    tempY = 0;
    temp = 0;

    //Populating the treasure array with bombs
    while (temp < (difficulty * difficulty))
    {
        tempX = randInt(difficulty * 4) - 1, 0;
        tempY = randInt(difficulty * 4) - 1, 0;
        if (treasureArray[tempX][tempY] != 'X' and treasureArray[tempX][tempY] != '@')
        {
            treasureArray[tempX][tempY] = '@';
            temp++;
        }
    }
    tempX = 0;
    tempY = 0;
    temp = 0;

    system("cls");
    while (running == 1)
    {
        printBoard(0, difficulty);
        while (validationLoop == 1)
        {
            printf("What would you like to do? Your choices are: \nCheck a space (1), Quit (2), View Key (3) \nYour choice: ");
            scanf("%d", &temp);
            if (temp > 0 and temp < 4)
            {
                break;
            }
            else
            {
                printf("Invalid Input! Choice must be between 1 and 3.\n");
            }
        }
    }
}

```

[illegible]

```

#pragma once
int randIn(int high, int low)
{
    return rand() % (high - low + 1) + low;
}

void printBoard(int board, int difficulty) //print the current state of the board to the console
{
    int tempX = 0;
    int tempY = 0;

    //printing the column numbers
    if (difficulty == 3)
    {
        printf(" ");
    }
    else
    {
        printf(" ");
    }
    while (tempX < difficulty * 4)
    {
        if (tempX < 9 and difficulty == 3)
        {
            printf("0%d ", tempX + 1);
        }
        else
        {
            printf("%d ", tempX + 1);
        }
        tempX++;
    }
    tempX = 0;
    printf("\n");
    //printing dividing line
    if (difficulty == 3)
    {
        printf(" ");
    }
    else
    {
        printf(" ");
    }
    while (tempX < (difficulty * 4) - 1)
    {
        if (difficulty == 3)
        {
            printf("-----");
        }
        else
        {
            printf("-----");
        }
        tempX++;
    }

    if (difficulty == 3)
    {
        printf("-----");
    }
    else
    {
        printf("-----");
    }
    tempX = 0;
    printf("\n");

    //printing the rows
    while (tempY < difficulty * 4)
    {
        //printing the row number
        if (tempY < 9 and difficulty == 3)
        {
            printf("0%d ", tempY + 1);
        }
        else
        {
            printf("%d ", tempY + 1);
        }

        //printing the row contents
        while (tempX < difficulty * 4)
        {
            if (difficulty == 3)
            {
                if (board == 0)
                {
                    printf("%c ", displayArray[tempX][tempY]);
                }
                else
                {
                    printf("%c ", treasureArray[tempX][tempY]);
                }
            }
            else
            {
                if (board == 0)
                {
                    printf("%c ", displayArray[tempX][tempY]);
                }
                else
                {
                    printf("%c ", treasureArray[tempX][tempY]);
                }
            }
            tempX++;
        }
        printf("\n");
        tempX = 0;
        tempY++;
    }
    printf("\n");
}

```

Original Code:

For this worksheet, there are several possible approaches to the problem.

Create your own test set to show the strengths and/or limitations of your solution. Use as many of the boxes below as you need.

Test: Difficulty Selection i. Valid input

Screenshot(s) of output:

Difficulty 1 / Easy

```
C:\Users\Samst_000\source\repos\Worksheet 5\Debug\Worksheet 5.exe
 1 2 3 4
-----
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: _
```

Difficulty 2 / Medium

```
C:\Users\Samst_000\source\repos\Worksheet 5\Debug\Worksheet 5.exe
 1 2 3 4 5 6 7 8
-----
1| 0 0 0 0 0 0 0 0
2| 0 0 0 0 0 0 0 0
3| 0 0 0 0 0 0 0 0
4| 0 0 0 0 0 0 0 0
5| 0 0 0 0 0 0 0 0
6| 0 0 0 0 0 0 0 0
7| 0 0 0 0 0 0 0 0
8| 0 0 0 0 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: _
```

```
Difficulty 3 / Hard
C:\Users\Samst_000\source\repos\Worksheet 5\Debug\Worksheet 5.exe
 01 02 03 04 05 06 07 08 09 10 11 12
-----
01| 0 0 0 0 0 0 0 0 0 0 0 0
02| 0 0 0 0 0 0 0 0 0 0 0 0
03| 0 0 0 0 0 0 0 0 0 0 0 0
04| 0 0 0 0 0 0 0 0 0 0 0 0
05| 0 0 0 0 0 0 0 0 0 0 0 0
06| 0 0 0 0 0 0 0 0 0 0 0 0
07| 0 0 0 0 0 0 0 0 0 0 0 0
08| 0 0 0 0 0 0 0 0 0 0 0 0
09| 0 0 0 0 0 0 0 0 0 0 0 0
10| 0 0 0 0 0 0 0 0 0 0 0 0
11| 0 0 0 0 0 0 0 0 0 0 0 0
12| 0 0 0 0 0 0 0 0 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice:
```

Result: Did the programme behave as expected ie syntactically and logically correct?
YES
The program produces a grid of the correct size.

Test: Difficulty Selection ii. Input out of range (4)

```
Screenshot(s) of output:
C:\Users\Samst_000\source\repos\Worksheet 5\Debug\Worksheet 5.exe
Welcome to the treasure hunt V1.1.
The game will randomly generate a grid. Treasure will be placed randomly on the grid.
Your task is to find the treasure. Enter the X and Y coordinates you want to check.
But be careful! There are bombs on the grid. Hitting the bomb means you lose! Good luck!

Key:
0 = Unchecked
@ = Checked, Empty
x = Checked, treasure
@ = Bomb (You don't want to see this!)

Select your difficulty!
1 = Easy(4x4, 2 treasure 1 bomb) 2 = Medium(8x8, 4 treasure 4 bombs) 3 = Hard(12x12, 6 treasure 9 bombs)
Your choice: 4
Invalid Input! Difficulty must be between 1 and 3.
Your choice:
```

Result: Did the programme behave as expected ie syntactically and logically correct?
YES
The input is rejected and the user is prompted to enter a different choice.

Test: Difficulty Selection iii. Leave blank

Screenshot(s) of output:

```
C:\Users\Samrt_000\source\repos\Worksheet 5\64\Debug\Worksheet 5.exe
Welcome to the treasure hunt V1.1.
The game will randomly generate a grid. Treasure will be placed randomly on the grid.
Your task is to find the treasure. Enter the X and Y coordinates you want to check.
But be careful! There are bombs on the grid. Hitting the bomb means you lose! Good luck!

Key:
0 = Unchecked
@ = Checked, Empty
X = Checked, treasure
@ = Bomb (You don't want to see this!)

Select your difficulty!
1 = Easy(4x4, 2 treasure 1 bomb) 2 = Medium(8x8, 4 treasure 4 bombs) 3 = Hard(12x12, 6 treasure 9 bombs)
Your choice:
-
```

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

YES

The cursor is moved to a new line but the program will refuse to continue until the user enters a valid input.

Test: Game menu i. Valid choices

Screenshot(s) of output:

1 / Check a space

```
E:\Downloads\Treasure Hunt V1.1.exe
 1 2 3 4
-----
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: 1
Enter the X (Horizontal) co-ordinate you want to check:
```

2 / Quit

```
Microsoft Visual Studio Debug Console
1 2 3 4
-----
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: 2

C:\Users\Sams000\source\repos\Worksheet 5\x64\Debug\Worksheet 5.exe (process 15892) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

3 / View Key

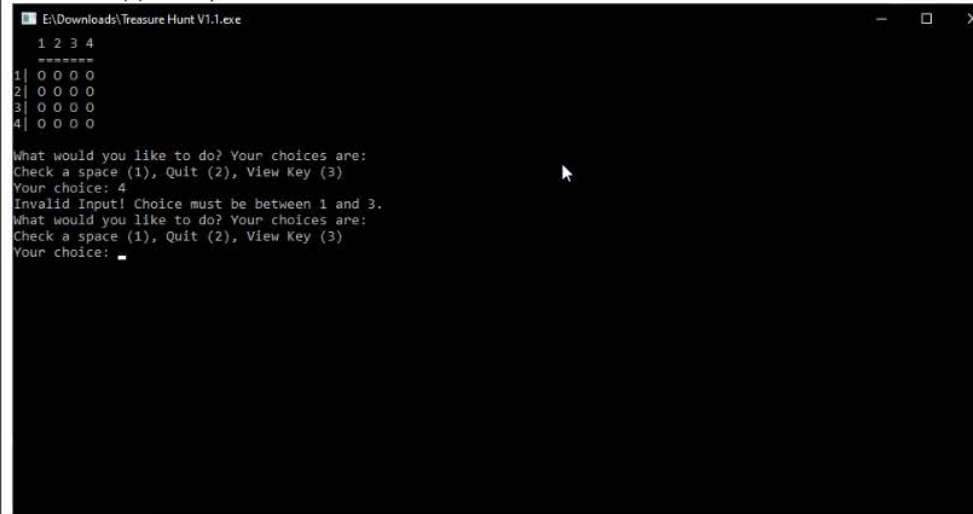
```
E:\Downloads\Treasure Hunt V1.1.exe
Key:
0 = Unchecked
@ = Checked, Empty
X = Checked, treasure
@ = Bomb (You don't want to see this!)

Type anything and hit enter to go back to the game.
_
```

Result: Did the programme behave as expected ie syntactically and logically correct?
YES
The program accepts the inputs and proceeds to the relevant screen.

Test: Game menu ii. Input out of range (4)

Screenshot(s) of output:



```
E:\Downloads\Treasure Hunt V1.1.exe
 1 2 3 4
-----
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: 4
Invalid Input! Choice must be between 1 and 3.
What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: _
```

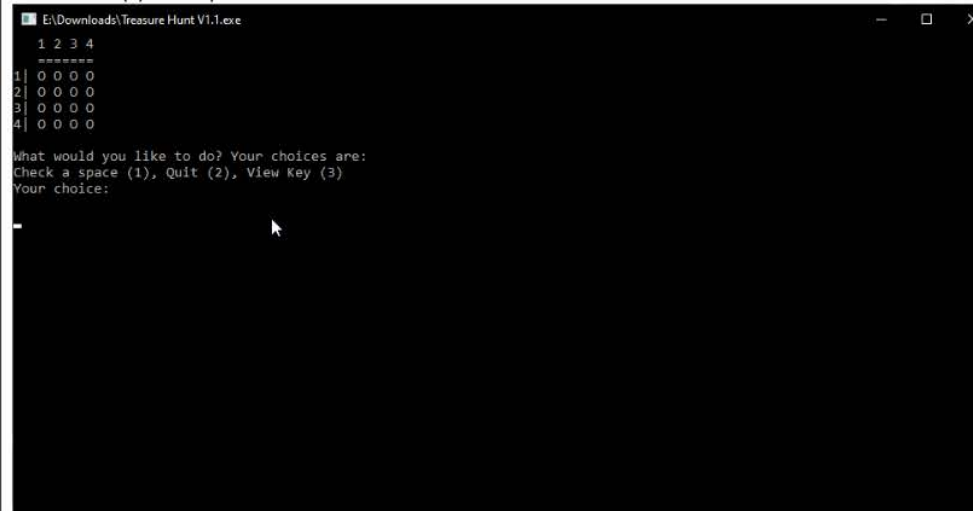
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

The program does not accept the input, displays an error and prompts the user to enter a valid input.

Test: Game menu iii. Leave blank

Screenshot(s) of output:



```
E:\Downloads\Treasure Hunt V1.1.exe
 1 2 3 4
-----
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice:
_
```

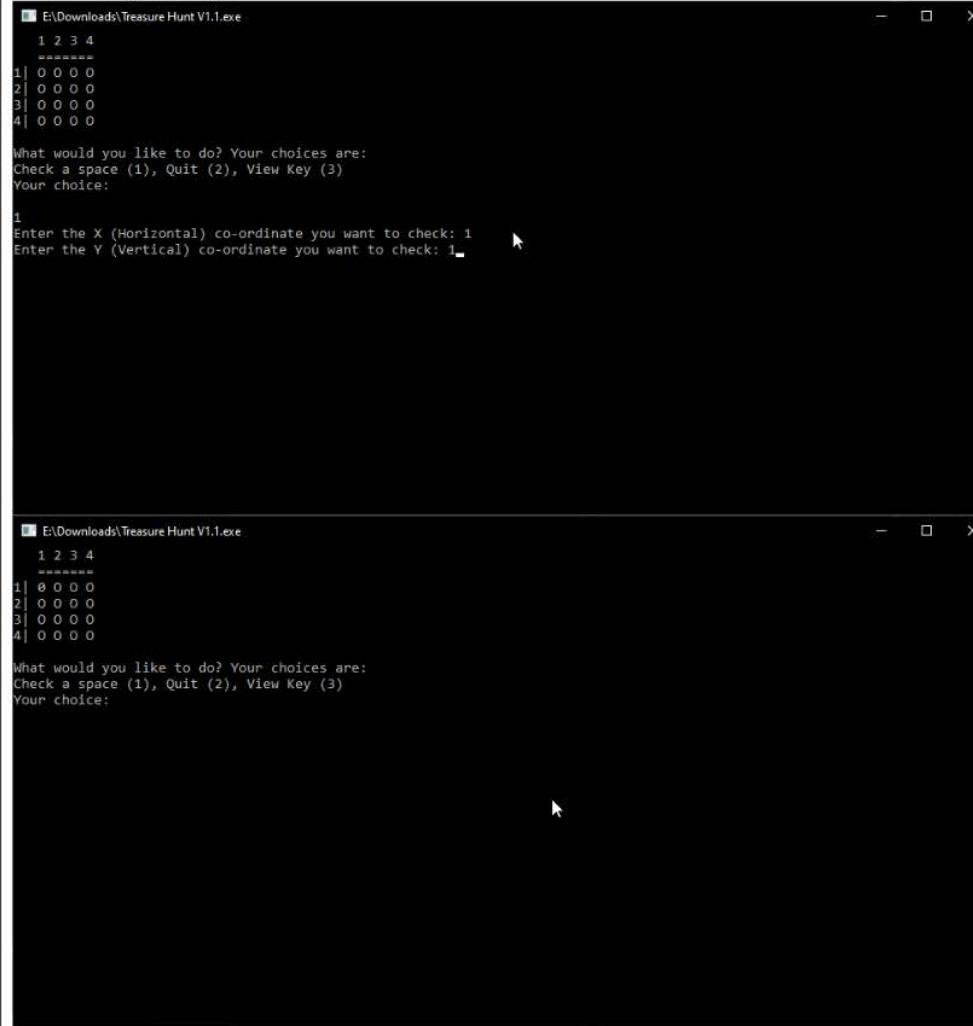
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

The cursor is moved to a new line but the program will refuse to continue until the user enters a valid input.

Test: Coordinates i. Valid entry

Screenshot(s) of output:



```
E:\Downloads\Treasure Hunt V1.1.exe
 1 2 3 4
-----
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice:
1
Enter the X (Horizontal) co-ordinate you want to check: 1
Enter the Y (Vertical) co-ordinate you want to check: 1

E:\Downloads\Treasure Hunt V1.1.exe
 1 2 3 4
-----
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice:
```

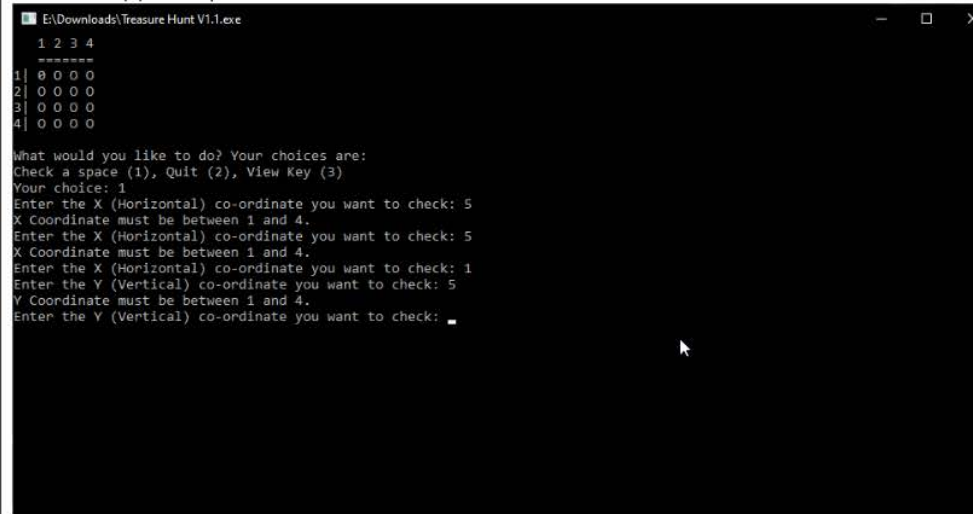
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

The correct position is changed to reveal its contents.

Test: Coordinates ii. Input out of range (5, 5)

Screenshot(s) of output:



```
E:\Downloads\Treasure Hunt V1.1.exe
 1 2 3 4
=====
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: 1
Enter the X (Horizontal) co-ordinate you want to check: 5
X Coordinate must be between 1 and 4.
Enter the X (Horizontal) co-ordinate you want to check: 5
X Coordinate must be between 1 and 4.
Enter the X (Horizontal) co-ordinate you want to check: 1
Enter the Y (Vertical) co-ordinate you want to check: 5
Y Coordinate must be between 1 and 4.
Enter the Y (Vertical) co-ordinate you want to check: _
```

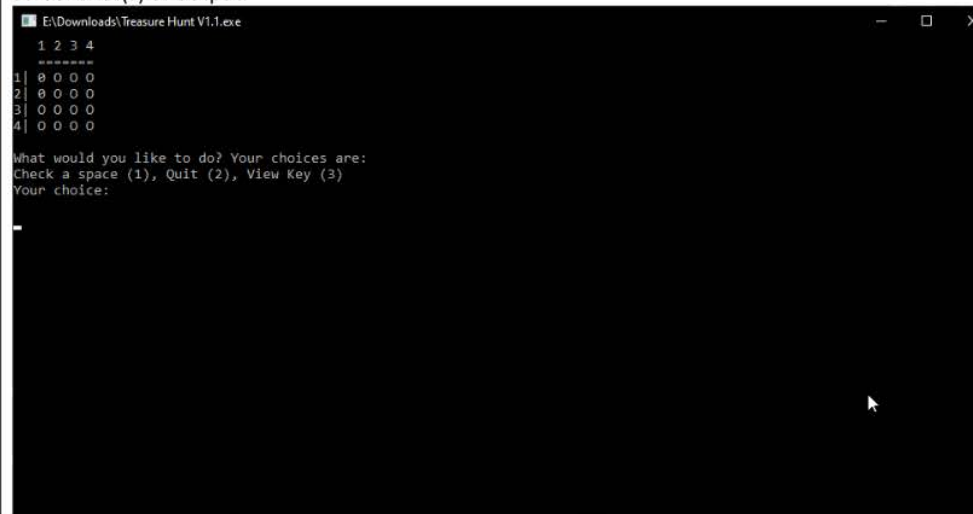
Result: Did the programme behave as expected ie syntactically and logically correct?

YES

The input is not accepted and the program does not continue

Test: Coordinates iii. Leave blank

Screenshot(s) of output:



```
E:\Downloads\Treasure Hunt V1.1.exe
 1 2 3 4
=====
1| 0 0 0 0
2| 0 0 0 0
3| 0 0 0 0
4| 0 0 0 0

What would you like to do? Your choices are:
Check a space (1), Quit (2), View Key (3)
Your choice: 1
Enter the X (Horizontal) co-ordinate you want to check:
Enter the Y (Vertical) co-ordinate you want to check: _
```

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

YES

The cursor is moved to a new line but the program will refuse to continue until the user enters a valid input.

Test: Win Condition (User finds all treasure)

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console

YOU WIN!

  1 2 3 4
  -----
1| X 0 0 0
2| 0 0 0 X
3| 0 0 0 0
4| 0 0 0 0

You found all the treasure. You win!
You found 3 pieces of treasure in 10 turns!

C:\Users\Samst_000\source\repos\Worksheet 5\x64\Debug\Worksheet 5.exe (process 30812) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

YES

The program displays the win screen. It displays the correct statistics for treasure found and number of turns. It displays the correct board state at the end of the game.

Test: Loss Condition (User hits a bomb)

Screenshot(s) of output:



```
Microsoft Visual Studio Debug Console

GAME OVER

  1 2 3 4
  -----
1| 0 0 X 0
2| 0 0 0 0
3| X 0 0 0
4| 0 0 0 0

You hit a bomb! Game over.
You lost after 2 turn(s).
You found 0 piece(s) of treasure.

C:\Users\Samst_000\source\repos\Worksheet 5\x64\Debug\Worksheet 5.exe (process 30904) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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

YES

The program displays the win screen, the entire board's contents and the correct statistics.

Final code:

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

char treasureArray[12][12];
char displayArray[12][12];

#include "Header.h"

void main()
{
    srand(time(NULL));
    int running = 1;
    int validationLoop = 1;
    int tempX = 0;
    int tempY = 0;
    int treasureFound = 1;
    int turns = 0;
    int difficulty = 0;
    char tempStr[90];

    printf("Welcome to the treasure hunt V1.2.!\n");
    printf("The game will randomly generate a grid. Treasure will be placed randomly on the grid. \n");
    printf("Your task is to find the treasure. Enter the X and Y coordinates you want to check. \n");
    printf("But be careful! There are bombs on the grid. Hitting the bomb means you lose! Good luck!\n");
    printf("a");
    printf("Key: a@ = Unchecked a@ = Checked, Empty aX = Checked, treasure a@ = Bomb (You don't want to see this!\n");
    printf("a");

    printf("Select your difficulty! \n1 = Easy(4x4, 2 treasure 1 bomb) 2 = Medium(8x8, 4 treasure 4 bombs) 3 = Hard(12x12, 6 treasure 9 bombs)\n");
    while (validationLoop == 1)
    {
        printf("Your choice: ");
        scanf("%d", &difficulty);
        if (difficulty > 0 and difficulty < 4)
        {
            break;
        }
        else
        {
            printf("Invalid Input! Difficulty must be between 1 and 3. \n");
        }
    }

    //Sets the arrays to blank spaces
    while (tempY < difficulty * 4)
    {
        while (tempX < difficulty * 4)
        {
            displayArray[tempX][tempY] = ' ';
            treasureArray[tempX][tempY] = ' ';
            tempX++;
        }
        tempX = 0;
        tempY++;
    }
    tempY = 0;

    //Populating the treasure array with treasure
    while (temp < difficulty * 2)
    {
        tempX = randint(difficulty * 4) - 1, 0;
        tempY = randint(difficulty * 4) - 1, 0;
        if (treasureArray[tempX][tempY] != 'X')
        {
            treasureArray[tempX][tempY] = 'X';
            temp++;
        }
    }
    tempX = 0;
    tempY = 0;
    temp = 0;

    //Populating the treasure array with bombs
    while (temp < (difficulty * difficulty))
    {
        tempX = randint(difficulty * 4) - 1, 0;
        tempY = randint(difficulty * 4) - 1, 0;
        if (treasureArray[tempX][tempY] != 'X' and treasureArray[tempX][tempY] != '@')
        {
            treasureArray[tempX][tempY] = '@';
            temp++;
        }
    }
    tempX = 0;
    tempY = 0;
    temp = 0;

    system("cls");
    while (running == 1)
    {
        printBoard(0, difficulty);
        while (validationLoop == 1)
        {
            printf("What would you like to do? Your choices are: \nCheck a space (1), Quit (2), View Key (3) \nYour choice: ");
            scanf("%d", &temp);
            if (temp > 0 and temp < 4)
            {
                break;
            }
            else
            {
                printf("Invalid Input! Choice must be between 1 and 3. \n");
            }
        }
        if (temp == 1) //checking a space on the grid
        {
            while (validationLoop == 1) //validation for X input
            {
                printf("Enter the X (Horizontal) co-ordinate you want to check: ");
            }
        }
    }
}
```



```

//print out
int randint(int high, int low)
{
    return rand() % (high - low + 1) + low;
}

void printBoard(int board, int difficulty) //print the current state of the board to the console
{
    int tempX = 0;
    int tempY = 0;

    //printing the column numbers
    if (difficulty == 3)
    {
        printf(" ");
    }
    else
    {
        printf(" ");
    }
    while (tempX < difficulty * 4)
    {
        if (tempX < 9 and difficulty == 3)
        {
            printf("%d", tempX + 1);
        }
        else
        {
            printf("%d", tempX + 1);
        }
        tempX++;
    }
    tempX = 0;
    printf("\n");
    //opening dividing line
    if (difficulty == 3)
    {
        printf(" ");
    }
    else
    {
        printf(" ");
    }
    while (tempX < (difficulty * 4) - 1)
    {
        if (difficulty == 3)
        {
            printf("==");
        }
        else
        {
            printf("==");
        }
        tempX++;
    }
    if (difficulty == 3)
    {
        printf("==");
    }
    else
    {
        printf("==");
    }
    tempX = 0;
    printf("\n");
    //printing the rows
    while (tempY < difficulty * 4)
    {
        //printing the row number
        if (tempY < 9 and difficulty == 3)
        {
            printf("%d", tempY + 1);
        }
        else
        {
            printf("%d", tempY + 1);
        }
        //printing the row contents
        while (tempX < difficulty * 4)
        {
            if (difficulty == 3)
            {
                if (board == 0)
                {
                    printf("%c", displayArmy[tempX][tempY]);
                }
                else
                {
                    printf("%c", treasureArmy[tempX][tempY]);
                }
            }
            else
            {
                if (board == 0)
                {
                    printf("%c", displayArmy[tempX][tempY]);
                }
                else
                {
                    printf("%c", treasureArmy[tempX][tempY]);
                }
            }
            tempX++;
        }
        printf("\n");
        tempX = 0;
        tempY++;
    }
    tempY = 0;
    printf("\n");
}

```

Worksheet 5 Testing

GRADEMARK REPORT

FINAL GRADE

/30

GENERAL COMMENTS

Worksheet 5:

Coding:

You've certainly taken the brief and run with it here! Well done. I am generally happy with your function work, although at times I think there is code repeated where it doesn't need to be ie could be a little more efficient.

14/15

Testing:

Quite extensive! Well done, I think you've got most possibilities here

14/15

Total: 28/30

Organisation of report, writing style, presentation of work:

Generally good - sometimes the code is very small and hard to read, do keep it consistent with the rest of your work.

8/10

Overall total for assignment: 90.5%

(unfortunately I have no choice but to cap the work to 40% due to late submission - a great shame as you've done very well. Avoid this in the future for other modules so that you are awarded marks that reflect your ability/effort)

Test mark: 82%

Overall mark for module: 53%

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

PAGE 14
