

GCSE

Computer Science (A452)

Controlled Assessment

(Practical Investigation into JavaScript)



Candidate Name: Abdullah Hussain

Candidate Number: 4136

Centre Number: 20283

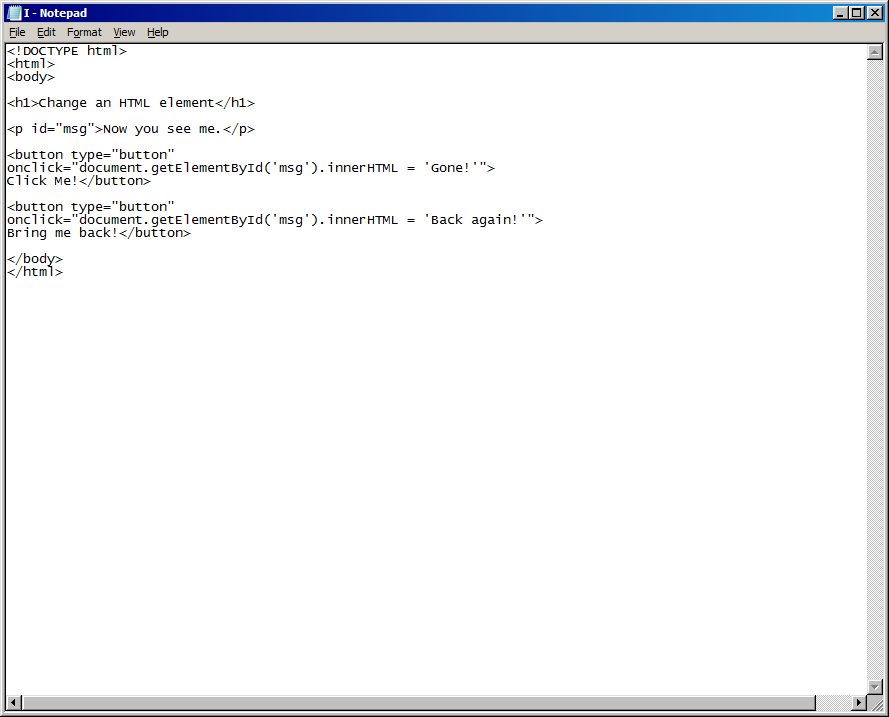
# Contents

1. Task 13
   1. Part I3
   2. Part II4
2. Task 28
   1. Part I8
      1. Planning of the code8
      2. Implementation of the code8
      3. Testing the code8
      4. Demonstrating the code9
   2. Part II9
      1. Planning of the code9
      2. Implementation and explanation of the code10
      3. Testing the code14
      4. Demonstrating the code15
3. Task 316
   1. Part I16
   2. Part II16
   3. Part III16
   4. Part IV17
      1. Code Explained17
4. Task 419
5. Task 50
   1. Benefits of saving JavaScript scripts in the HTML webpages20
   2. Drawbacks of saving JavaScript scripts in the HTML webpages20
   3. Benefits of saving JavaScript scripts externally as script files20
   4. Drawbacks of saving JavaScript scripts externally as script files21
6. Bibliography21

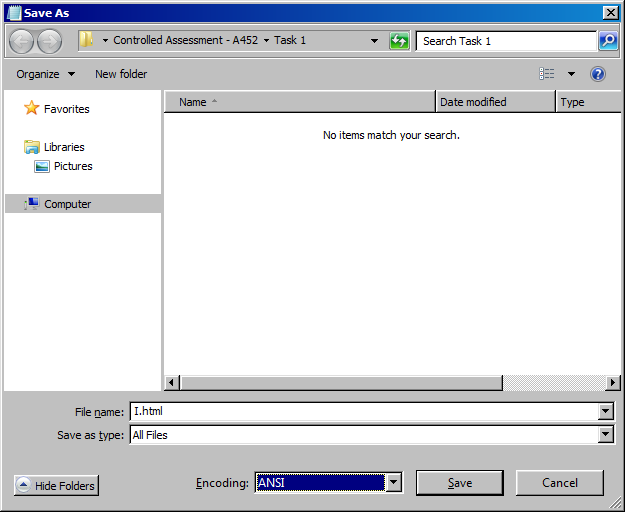
## Task 1

### Part I

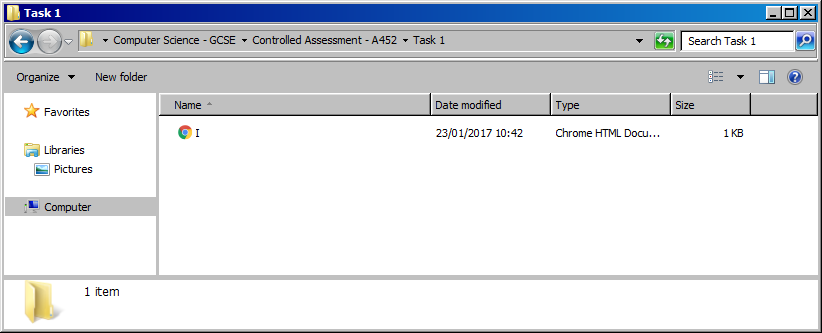
The script for task 1 part I was simply ran by copying the source script, which was provided by the controlled assessment, into notepad and saving the file extension as “.HTML”, see below. This is done so that it assigns the file the web format and thus can be interpreted by most modern web browsers, see below for the webpage that the script outputs. They are file formats that are based on “.txt” file formats and contain the layout for the representation of webpage content. For this reason, it can then be edited again in notepad. Alternatively I could’ve placed the source script in Adobe Dreamweaver; however, it uses up more resources and is not needed in this context.

Copying the source script into Notepad

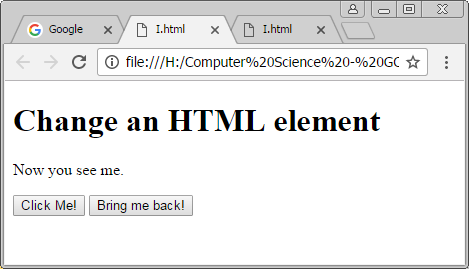
Saving the file as a .html



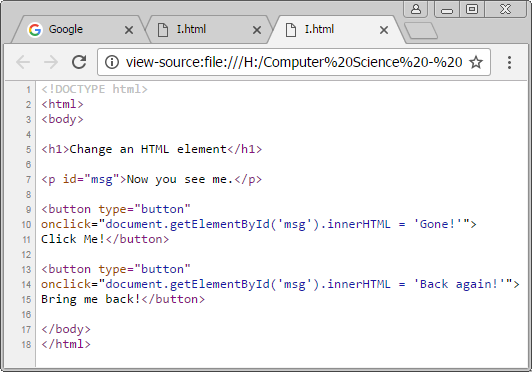
The saved file in a folder



What you see when you open the .html file



### Part II



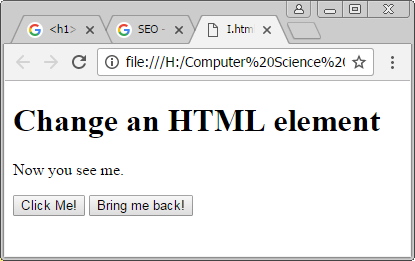
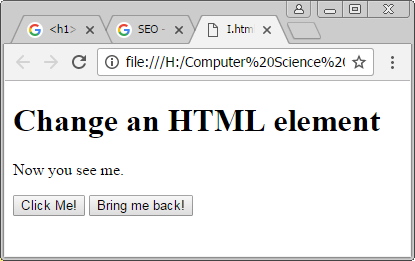
On **line 1** we can find the “<!DOCTYPE html>” declaration. Its sole responsibility in the script is to inform the web browser what version of HTML the page is written in. It is important to note that this declaration is not a tag but simply an instruction.

On **line 2** the “<html>” tag is used to tell the browser that this file is a HTML file. This tag is often referred to as the main root or the root element and this is because it is the base and everything encloses it. It is the parent attribute and everything else is the descendant of it or the child attribute.

On **line 3** the “<body>” tag or element is used to define the main content of the HTML document that will display the contents of the website from pictures to text to hyperlinks. You can use CSS to style the body and its contents like font colour etc.

On **line 5** there is a header, which contains the text “Change an HTML element”, which is assigned/defined using the “<h1>” tag. It is the heading tag which displays a heading, which is a large text title, and is used to structure the webpage and to index its content and structure. There are different variants all the way from H1 to H6 decreasing in text size and overall importance. You can also horizontally align the header using the align attribute. The header is ended by using the “</h1>” tag that closes the header. Furthermore, the header helps with SEO (Search Engine Optimisation) which helps to publicise and easily find your webpage when using a search engine.

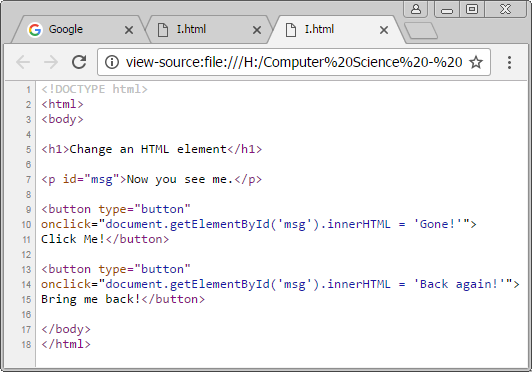
This is the header text that is defined in the source script and it’s the first thing that comes up and the largest due to the h1 tag.



This is the paragraph text, which is specifically chosen in the source script that is output using the p tag.

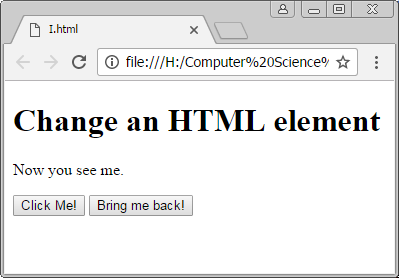
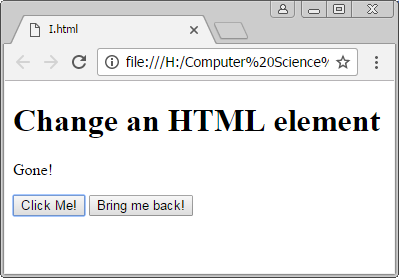
On **line 7** the “<p>” tag is the element that defines a paragraph of text, in this case “Now you see me.” The text must be placed in quotation marks so that it is recognised as a string. The paragraph is a block level element that occupies the entire space of its container as well as having a newline before and after. This paragraph is ended using the closing “</p>” tag.

* In this case within the “<p>” tag, there is an id attribute, called “msg”, that helps to manipulate the contents of the tag. It does this by assigning a unique identifier, by using the assignment operator “=”, for a HTML element so that JavaScript can access it and change the elements via HTML DOM, which is a standard for accessing documents.



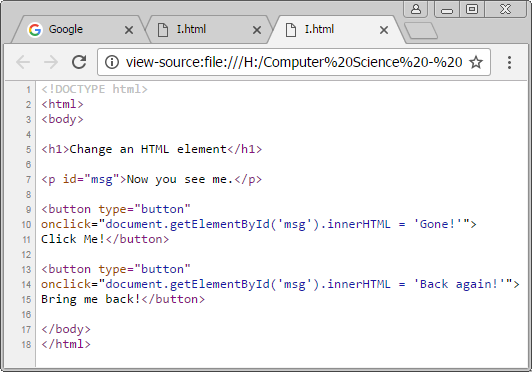
On **line 9** the script uses the <button> tag which defines a clickable, interactive button. Within the opening tag, the script has assigned the “button” attribute to the type. This allows the button to be clicked for something to happen. Other types include submit which provides the script with information and reset which resets the data. The button tag contains events, methods and properties but it also contains the text that will be displayed on the button, see **line 11**, in this case “Click Me!”. On line 11 the script ends with an ending tag “</button>”.

* On **line 10** the script, within the button tag, uses the onclick Event which causes an event to occur only if the user clicks the button element. The onclick Event contains the “document.getElementById(“msg”)” method that calls the contents of the element, in this case the text in the paragraph tag, that is assigned that specific id, in this case its “msg”. Furthermore the method previously stated is also embedded within the “.innerHTML” property which is assigned a text “Gone!”. This property changes the contents of the id with the content that is assigned to it. In this instance it replaces the content of the paragraph “Now you see me.” With the contents of the “.innerHTML” property which is “Gone!” once the button, that all of this is assigned to, is clicked.

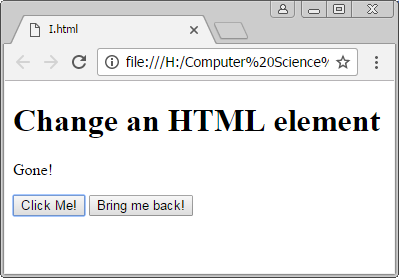
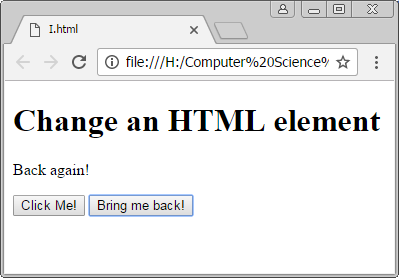
 

Before clicking the first button. After clicking the button.

On the image on the left is the paragraph text that is assigned a unique id. This paragraph, once the first button is clicked, will change to the text shown on the right image by using the .innerHTML property.



**Lines 13, 14 and 15** have the same function as line 9, 10 and 11 except with different content. These lines of script creates another clickable button that, once again, gets the contents of the paragraph element, by using the “document.getElementById(“”)” method and calling the id of the paragraph and continuing to embed the “.innerHTML” property which, once again, replaces the contents of the paragraph with the text that the property is assigned, in this case “back again!”.

Before clicking the second button After clicking the button.

On the image on the left is the paragraph text that is assigned a unique id. This paragraph has already changed once and will change to the text shown on the right image by using the .innerHTML property, when the second button is clicked.

but after clicking the first.

On **line 17 and 18** the ending tags close the body tag, by using the “</body>” tag, and the html tag, by using the “</html>” tag.

## Task 2

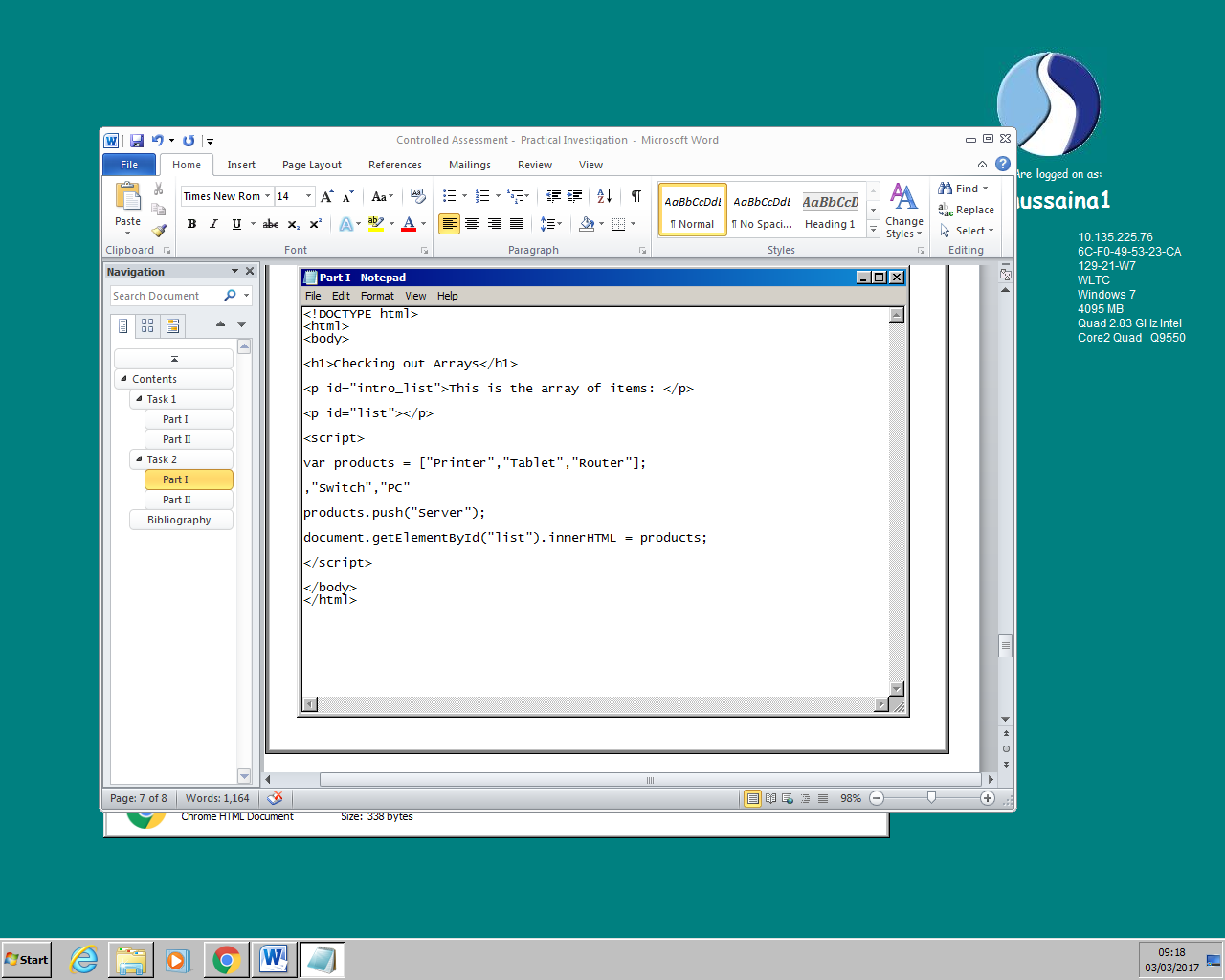
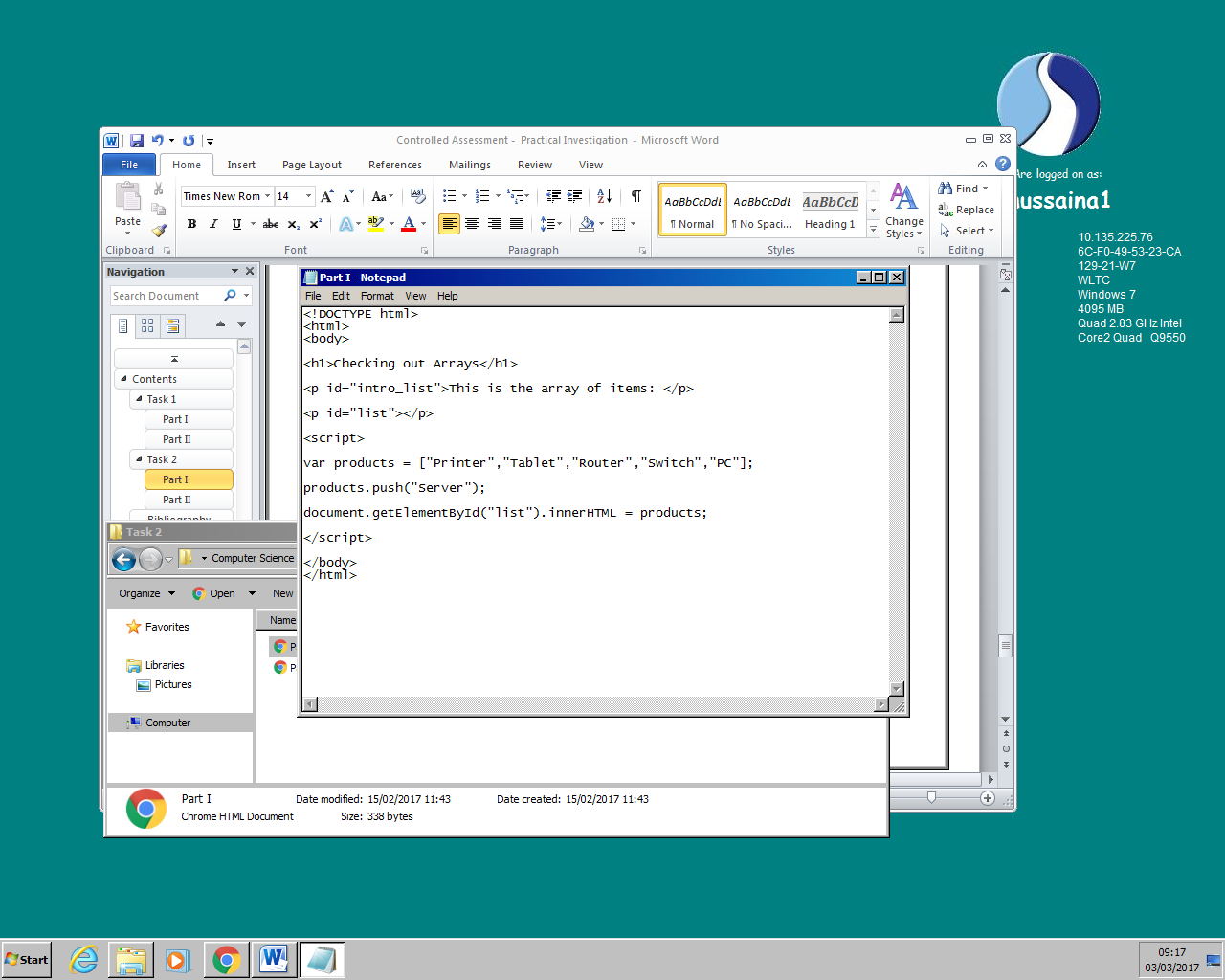
### Part I

#### Planning of the code

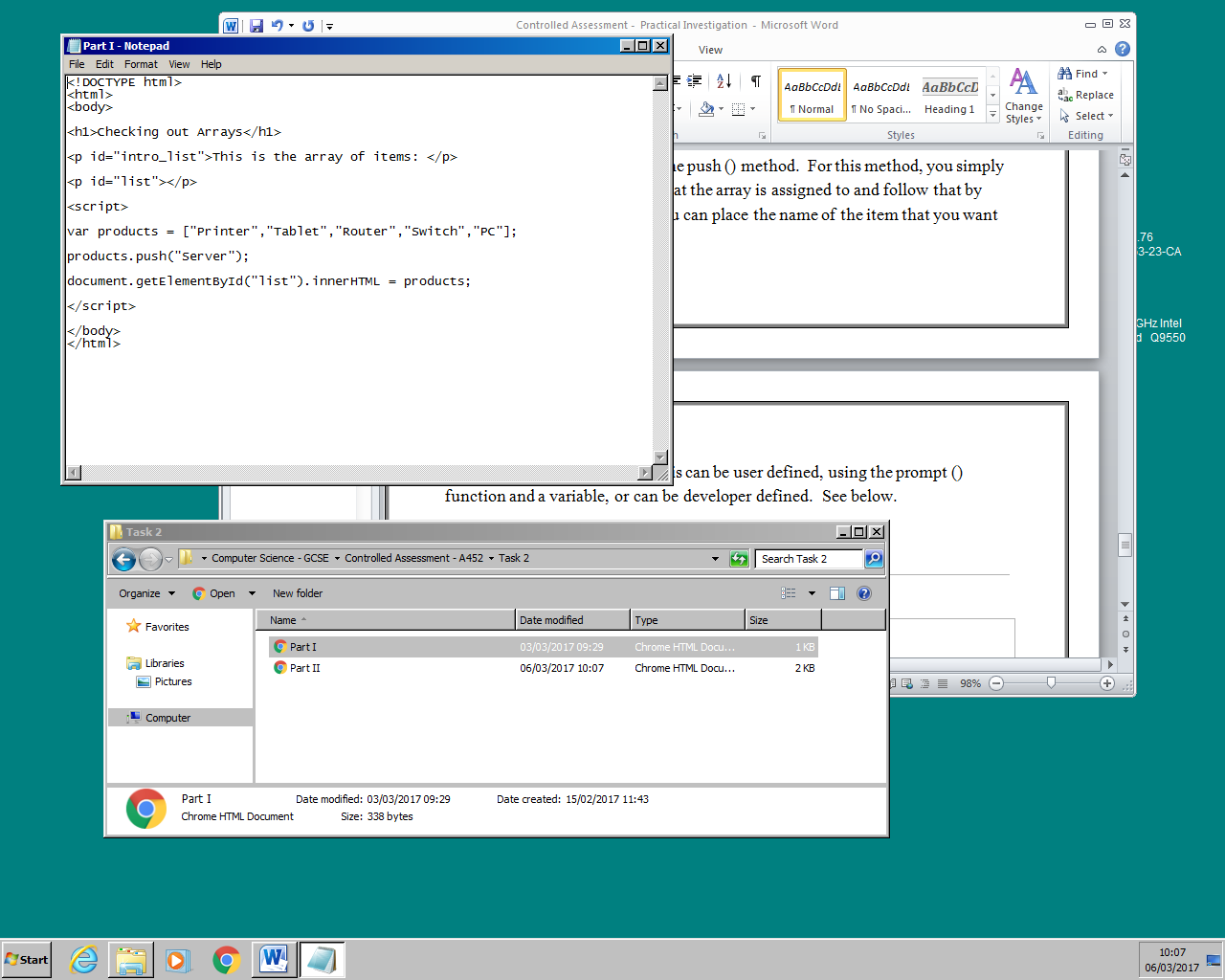
The task requires the coder must set up an array that contains some of the items provided and allows the coder to add their own items. This can be done in two ways. The first being to simply add an item by placing a comma after the last item and placing another item after it in quotation marks. The second way is that you can use the .push () method to add something which is a more dynamic way of adding an item. This method can be used to add a user defined item, which this script also does, using the prompt () function.

#### Implementation of the code

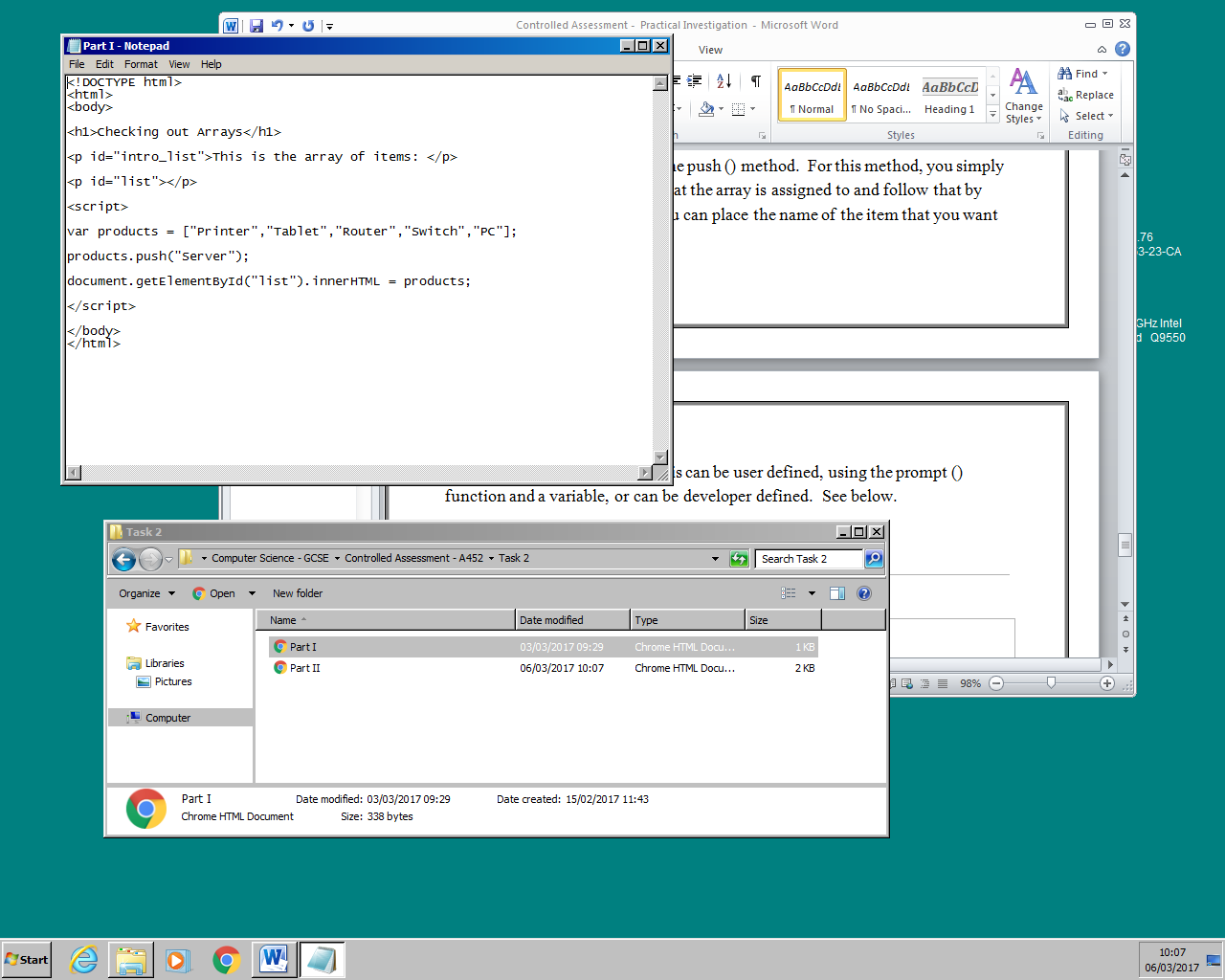
The first method:

The second method:



The second method that allows user input:



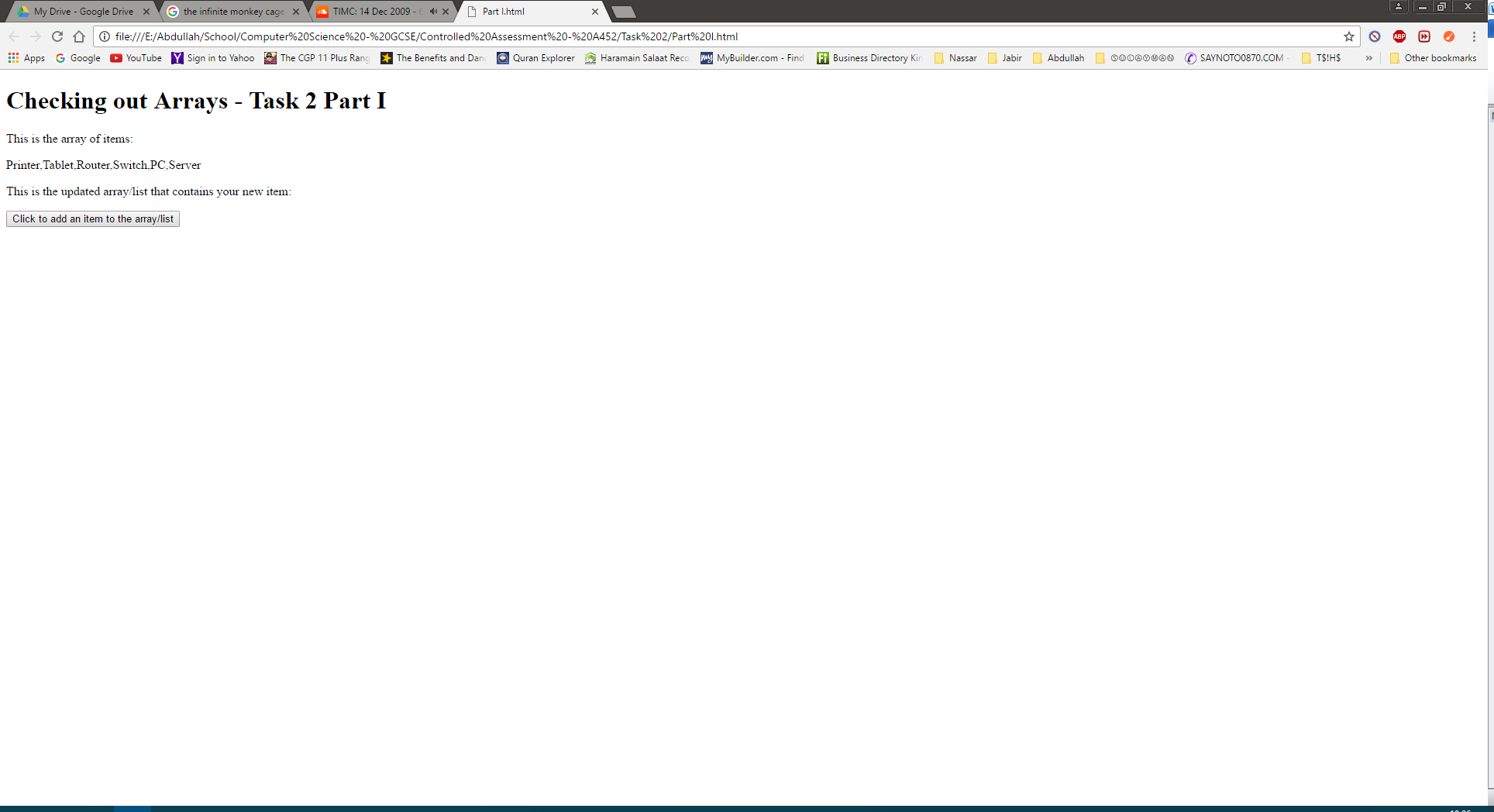


#### Testing the code

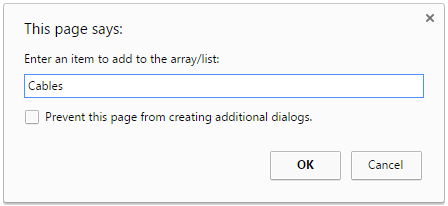
|  |  |  |
| --- | --- | --- |
| Criteria/Function tested: | Findings: | Success/Failure: |
| The array must include items provided and other items. |  | Success |
| Allows the user to enter their own items |  | Success |

#### Demonstrating the code

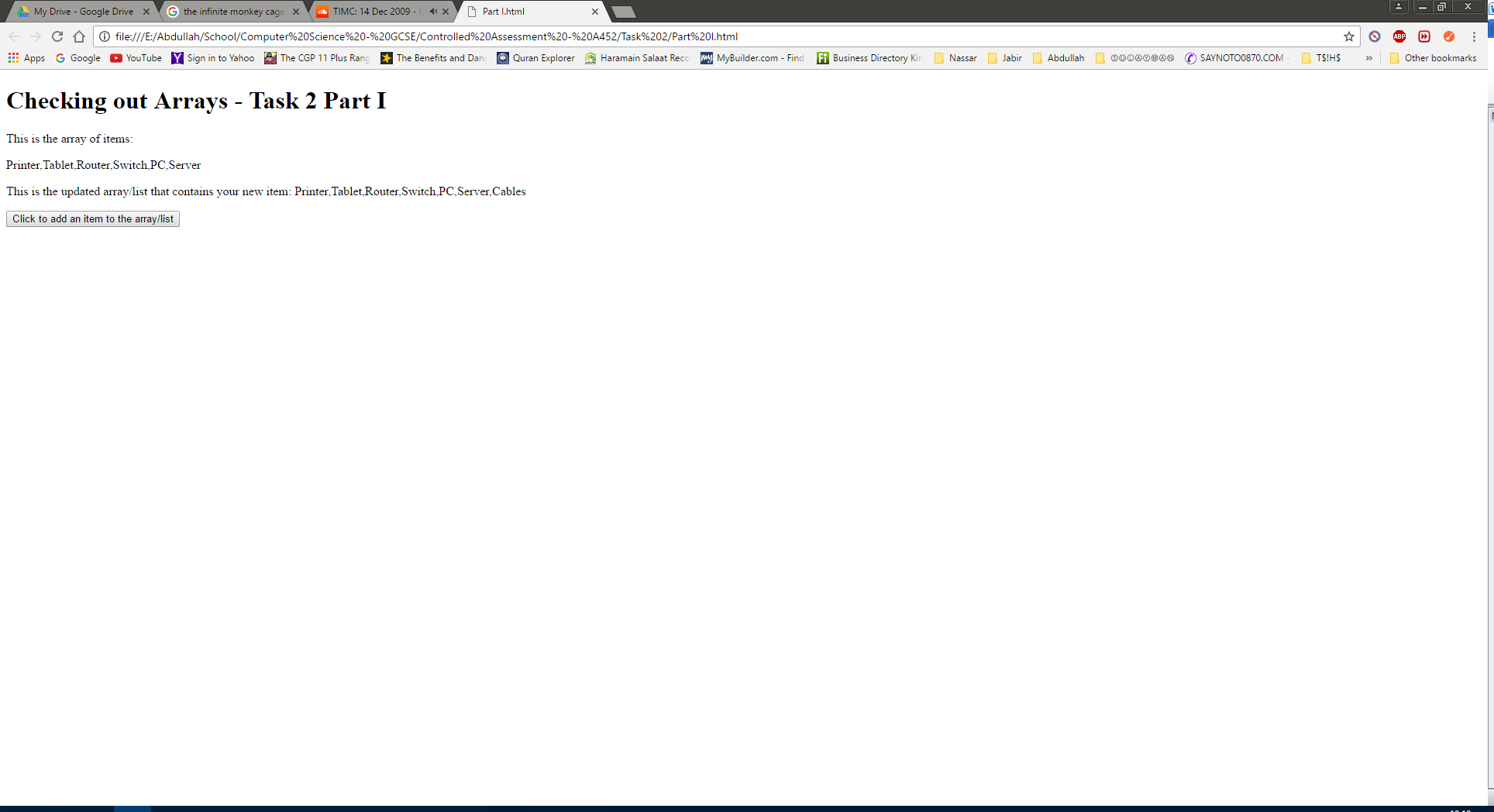
When you first open the web page you are greeted with a list of items and the option of adding your own item, see below.



If you click the button, it allows the user to enter an item, see below.



If you add an item, it will be added to another list below the original, see below.



### Part II

#### Planning of the code

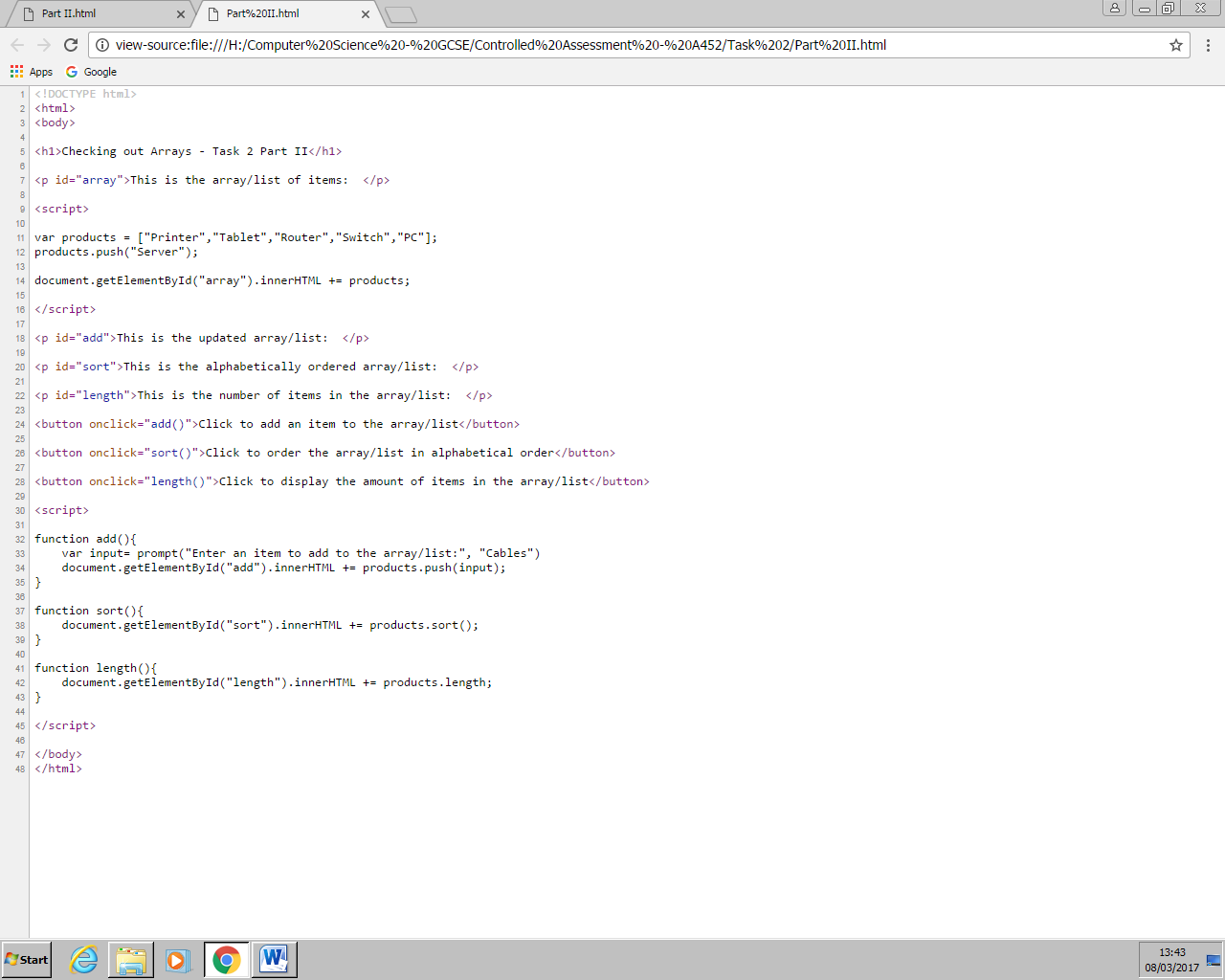
The task requires the coder to write a script that outputs the items in the array, made in the previous task, in alphabetical order and the number of items in the array.

The ordering of the items in the array is done by using the sort () method whilst finding the number of items in the array is done by using the .length property. Once again the code allows the user to place an item in the list and order it or find the number of items in the list.

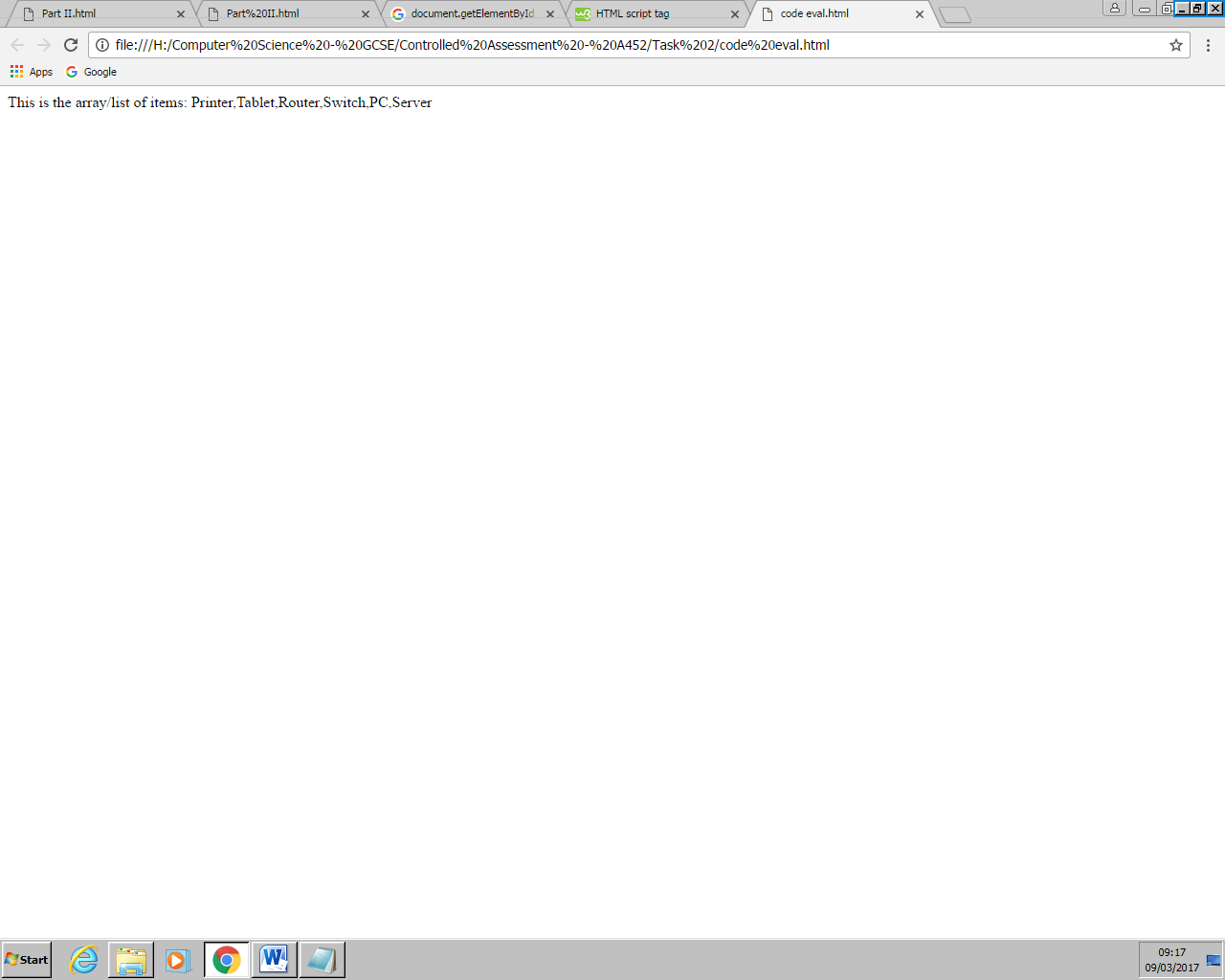
#### Implementation and explanation of the code

This task requires the coder to write a script that outputs the items in the array, in the previous task, in alphabetical order as well as output the number of items in the array.

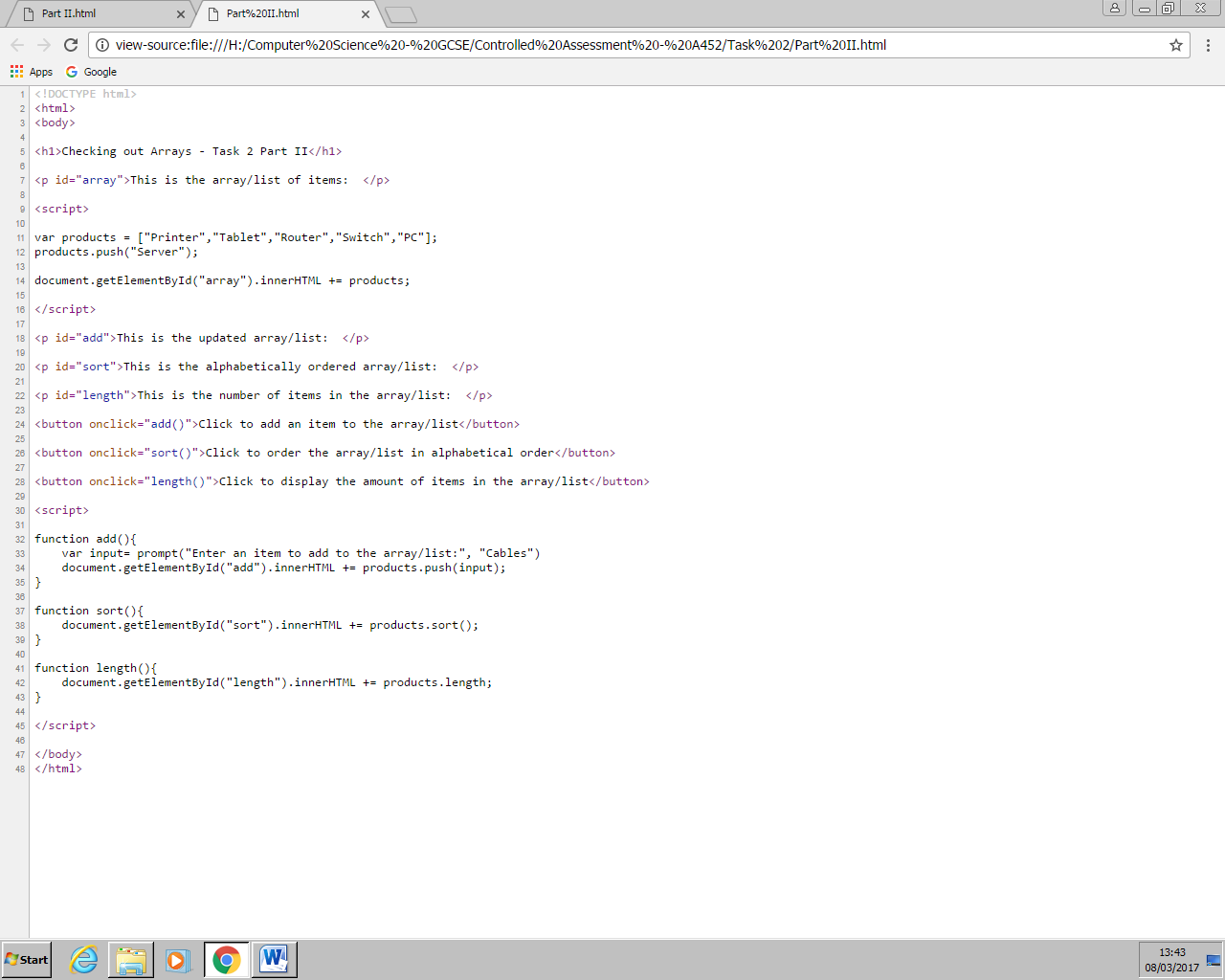
The first thing to do was to create an array with the items that make up the unordered list. To do this simply use a variable and place the items, in squared brackets, separated by commas. See **line 11** below. We can also add to the array by using the push () method. See **line 12** below. We then have to create a paragraph tag so that we can return the value of the array to it, by using the document.getElementById method, see **line 7**. The paragraph tag will be assigned the id “array”, see **line 7**, so that we can call it with the document.getElementById method. This method, in this case, simply calls the id of the paragraph tag while the .innerHTML property replaces the contents of the <p> element with what the property is assigned to, in this case the contents of the variable “product”, see **line 14**. To assign the content to the paragraph, in this case we appended it by using the “+=” operator, see **line 14**. This means it adds the content to whatever is in the paragraph tag, in this case the introductory phrase. The whole of the script that displays the contents of the list is enclosed in the opening <script> tag and the closing </script> tag, see **line 9 and 16**. These tags define the script within it as client-side.



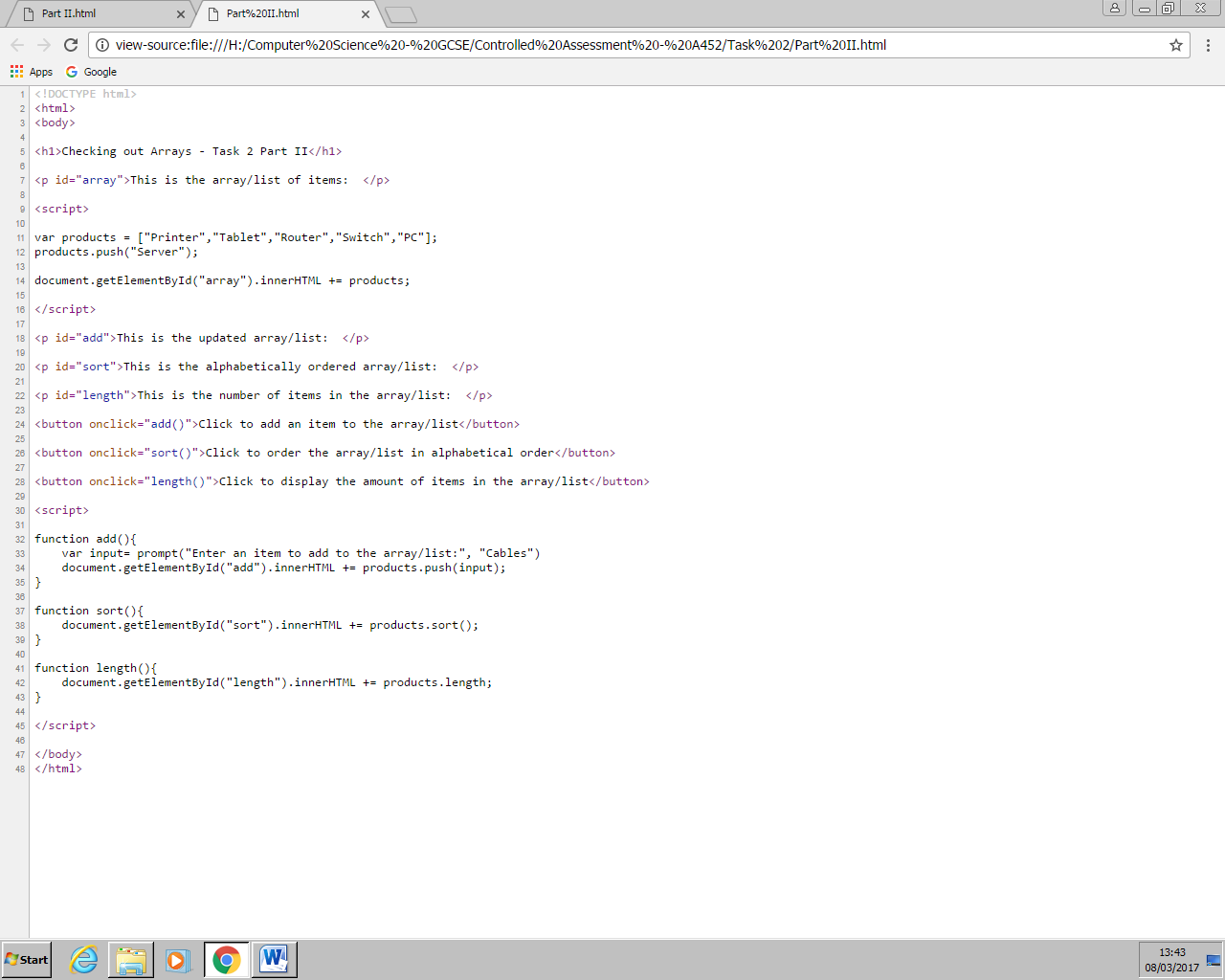
This segment of the script, from **line 7 to 16,** simply displays the introductory sentence and the items in the array, in the web browser. See the picture below.



We then had to delve into completing the task in hand; ordering the items in the list. To do this, once again we have to make a paragraph element, in this case with the id “sort”, so that we can return the ordered list to it. To do this, I created a button and a function so that this is possible. The button used the onclick event. This event calls is the event that happens when the button is clicked. In this case it activates the function sort().

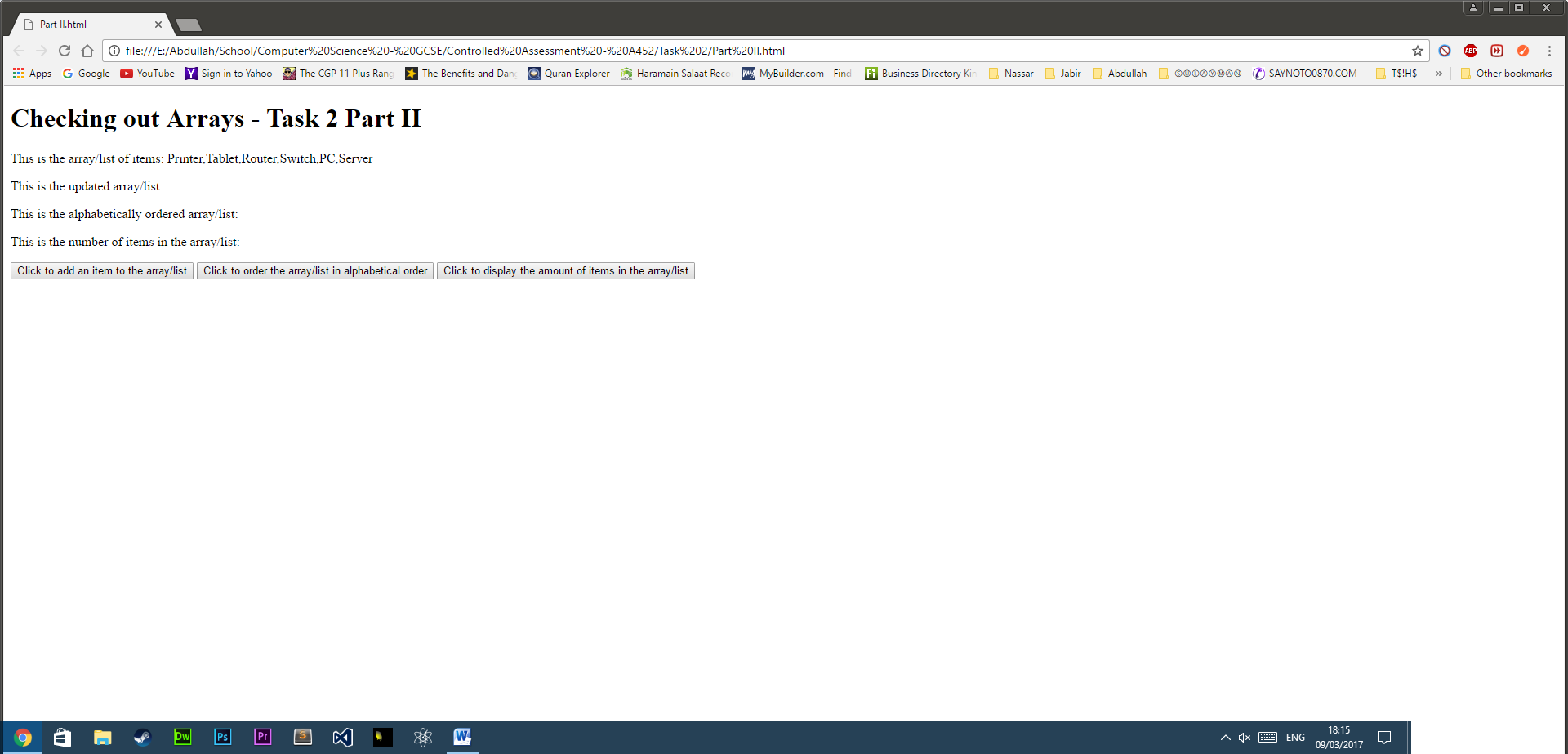


The next step is to start to order the list. This is done by declaring a function called “sort ()”, which was defined on **line 37.** It contains the document.getElementById() method as well as the innerHTML property, **see line 38**. The method calls the paragraph element, with the id “sort”, and then, the property, returns the sorted list by manipulating the array using the sort() method. This method is used by writing the variable name, placing a full stop and then typing the methods name. Consequently, it sorts the items, in the previously made array on **line 11**, in alphabetical order.

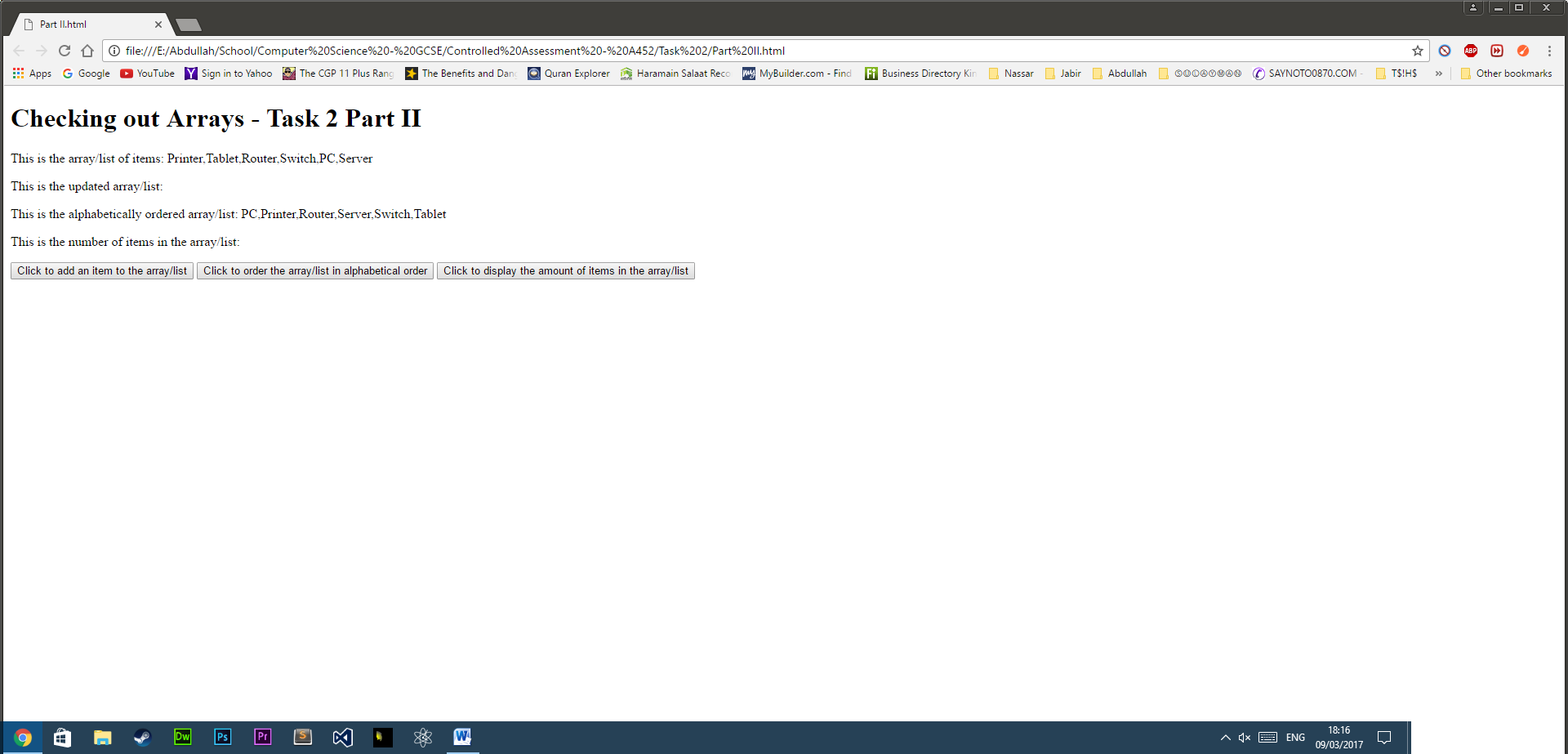


To make this process more user friendly, a button was added. This button was defined using the button tag, on **line 26**, and was assigned an onclick event that invokes the “sort ()” function. The button says to “Click to order the array/list in alphabetical order” to prompt the user to click the button.

The script, including the paragraph element, the button (with the onclick event) and the function (that sorts the array), all come together to display an introductory sentence, see images below, to show where the ordered list is and a button for the user to click to display the ordered list, see images below.

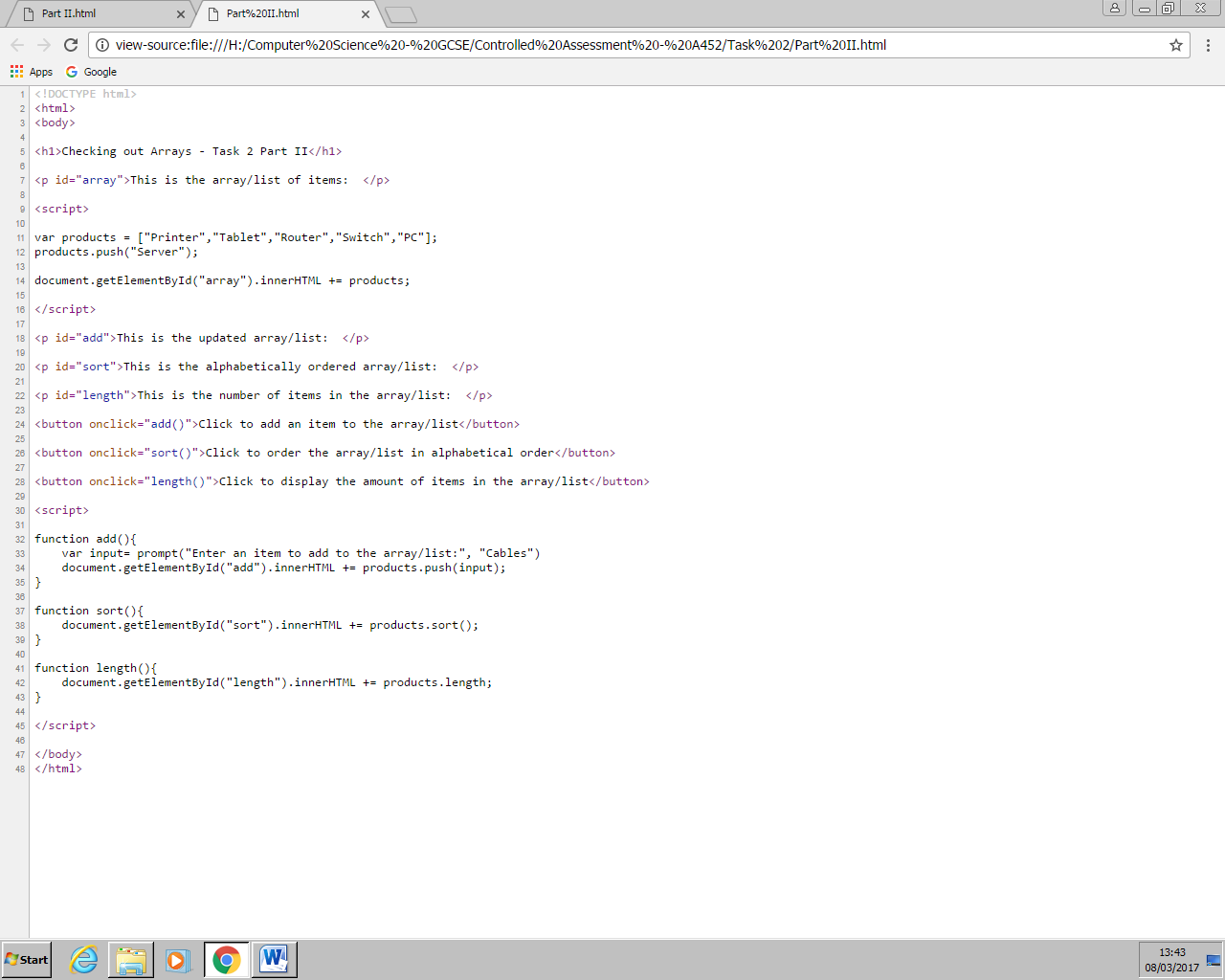


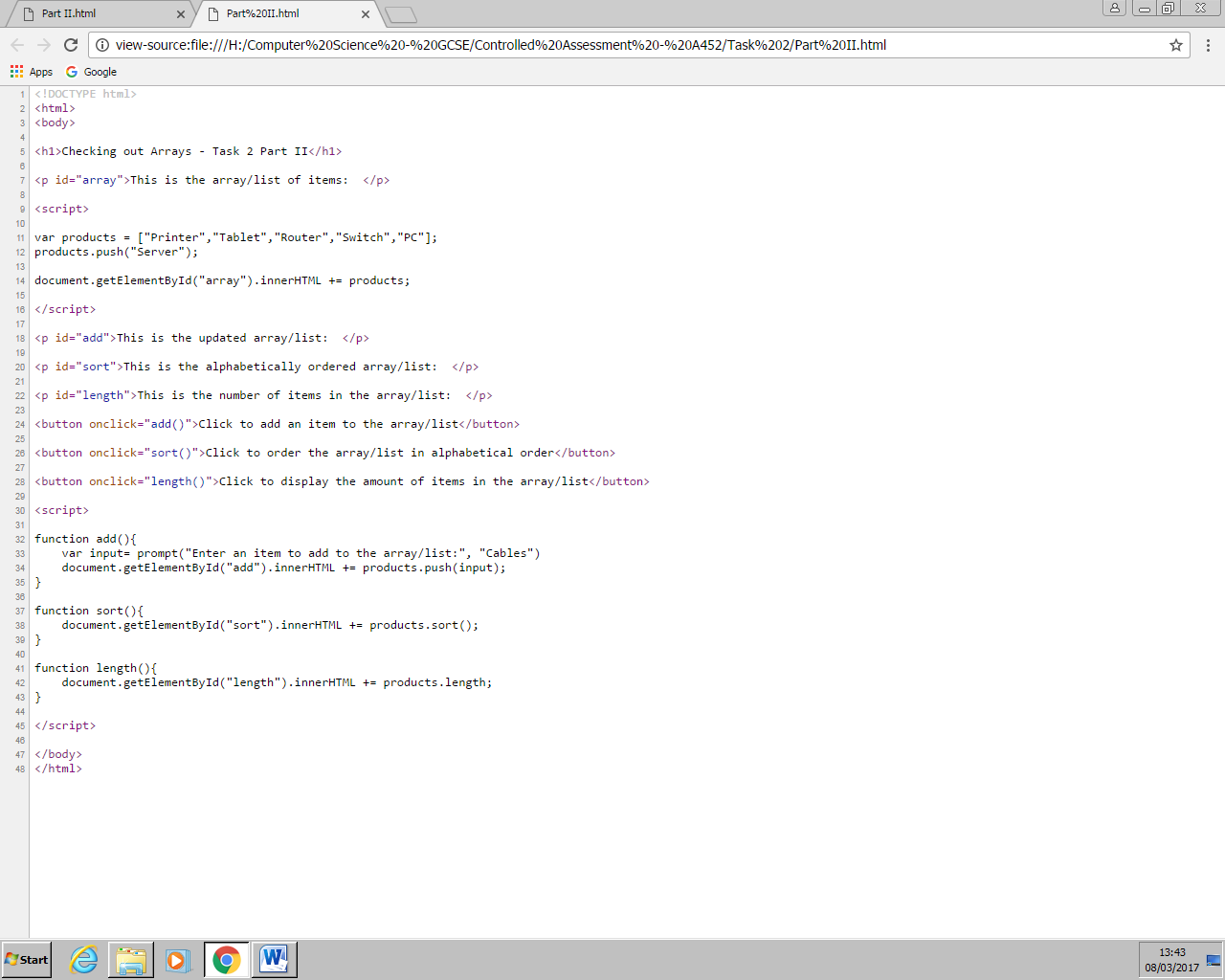
The script displays the introductory sentence only, when the button hasn’t been clicked.

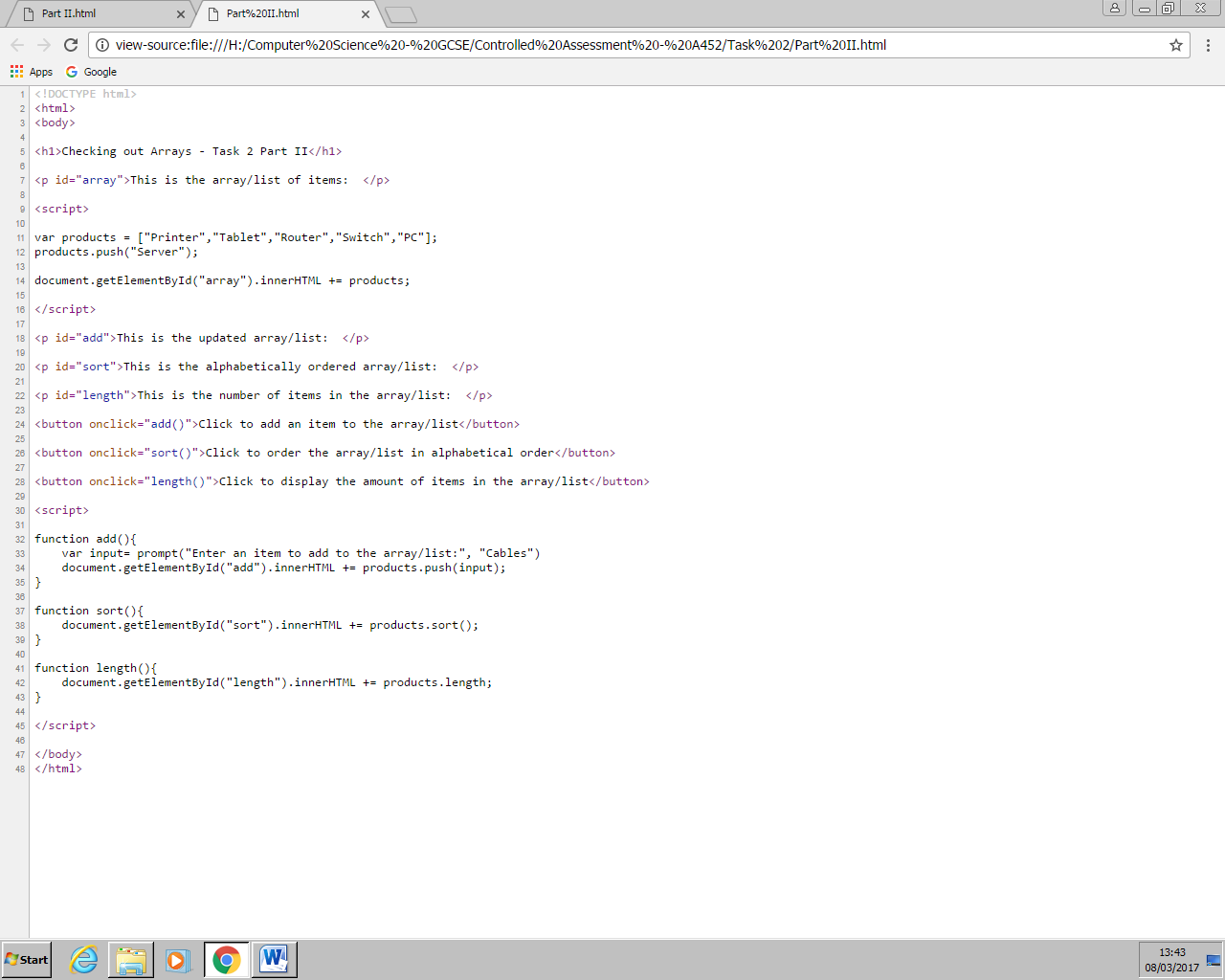


After the button was clicked, it displays the introductory sentence as well as the ordered list.

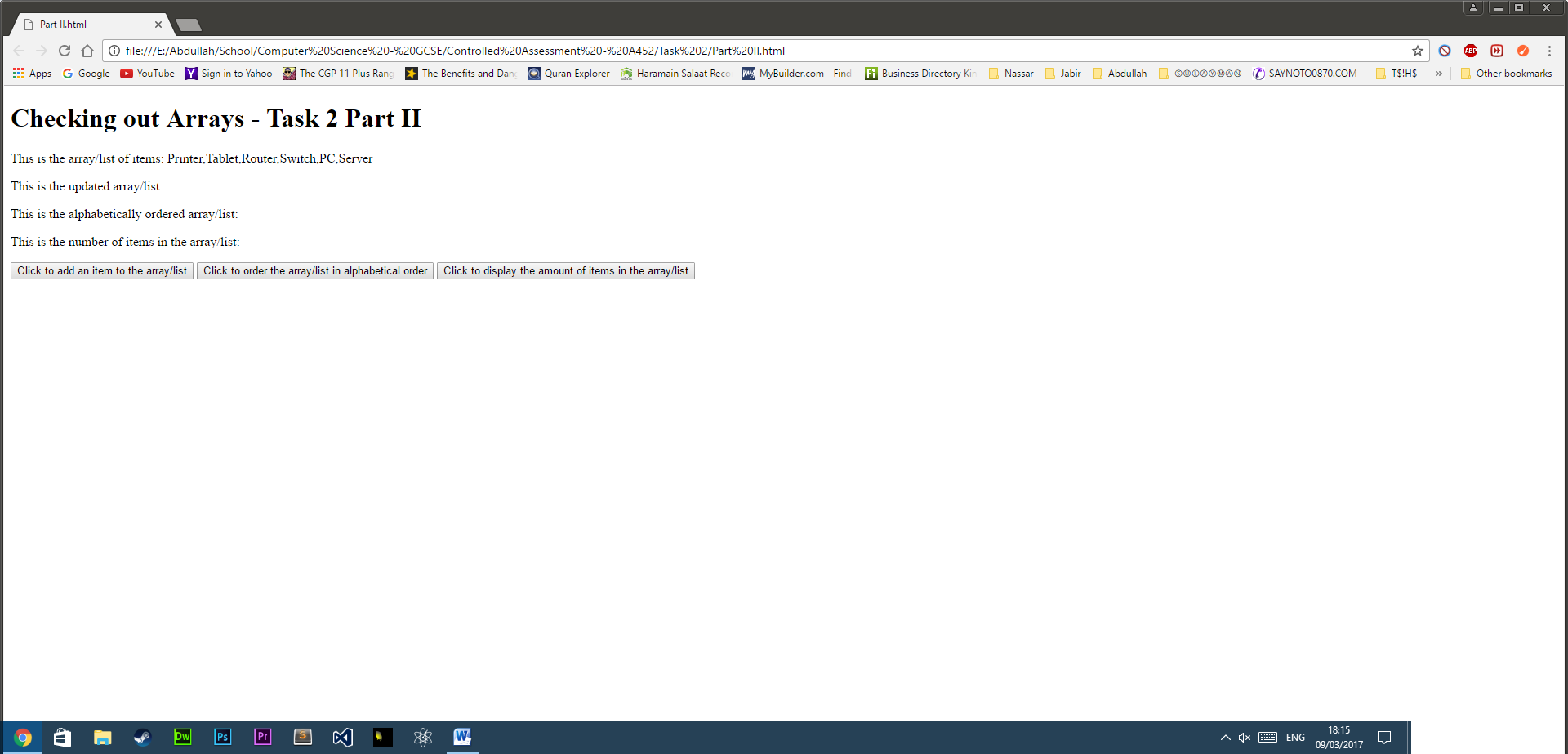
The next phase was to create a script that counted the number of items in the array/list. This, similar to with the sorting of the items, will be done by first defining a paragraph and assigning it an id, in this case “length”, see **line 22**. You then create a button, for a user friendly experience, that invokes the function “length ()” by using the onclick event, see **line 28**. Finally, we create a function, again with the name “length”, see **line 41**, that counts the number of items in the array. Within the function, the script calls the paragraph element, by using the document.getElementById method, and then appends to its contents the variable that the innerHTML property is assigned to, in this case products. However, the products variable is attached to the .length property, see **line 42**, which returns the number of items in the array to the paragraph element.



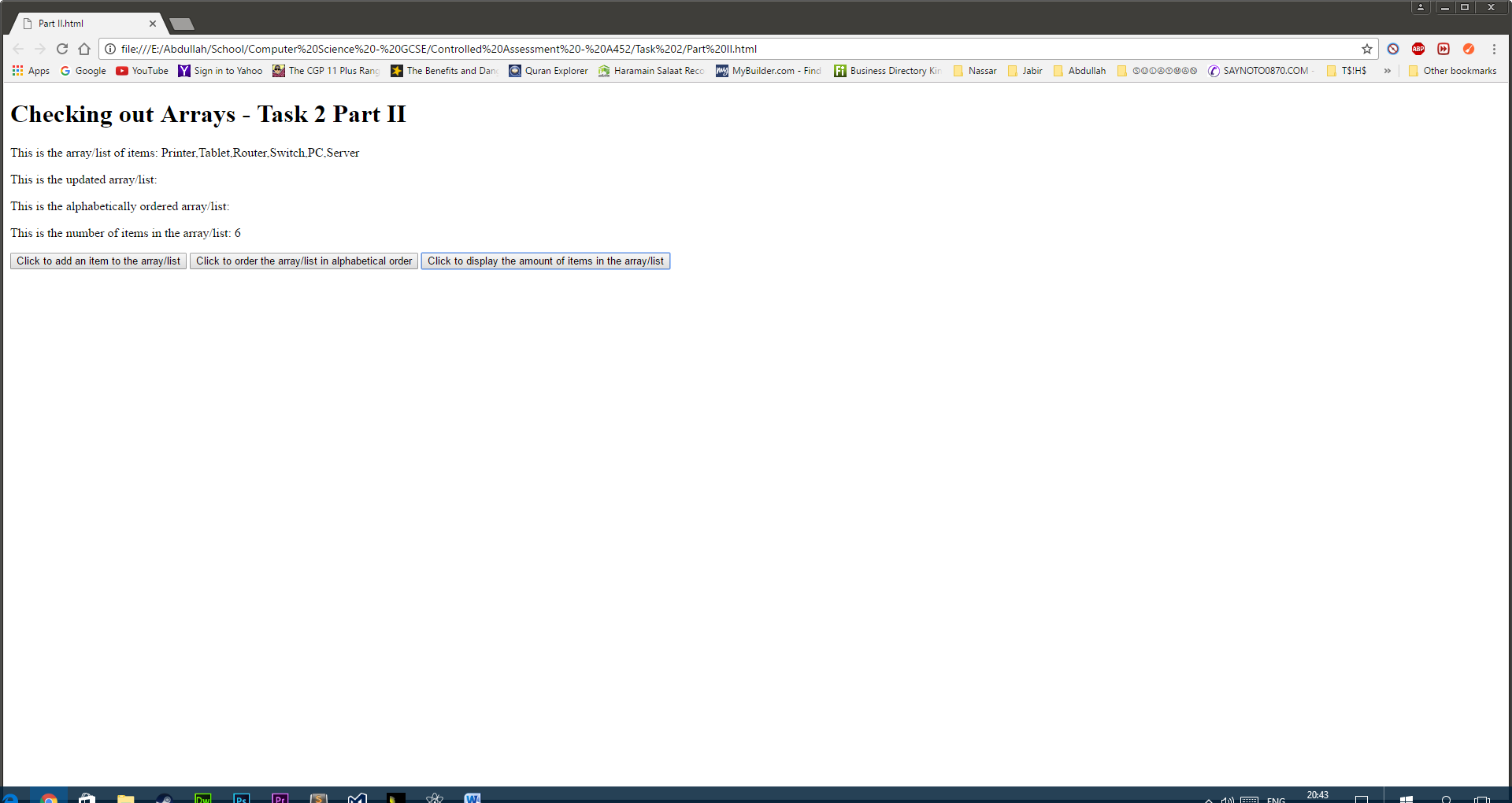




The script comes together to display an introductory sentence, see images below, to show how many items in the list and a button for the user to click to display the number of items, see images below.

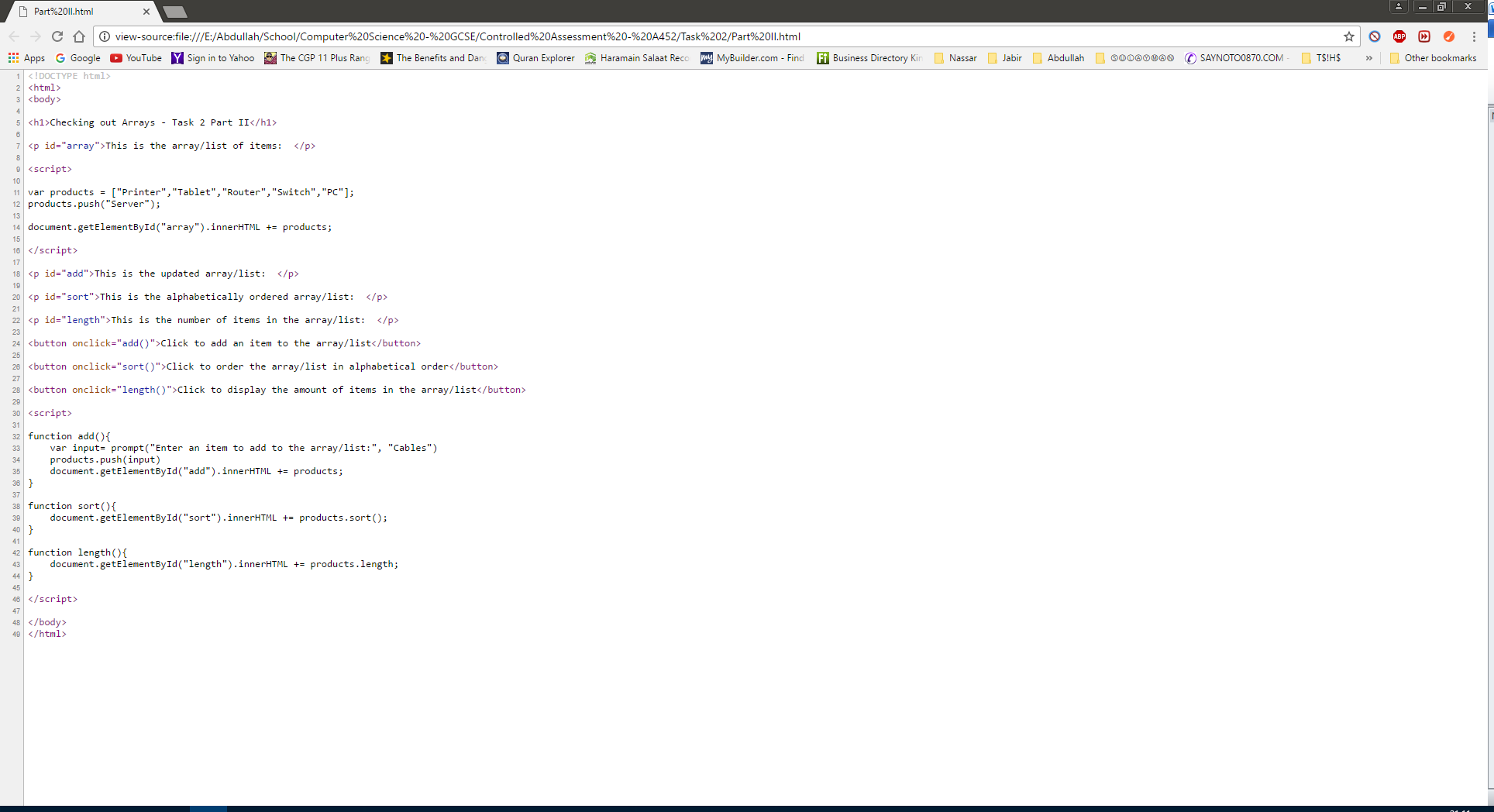


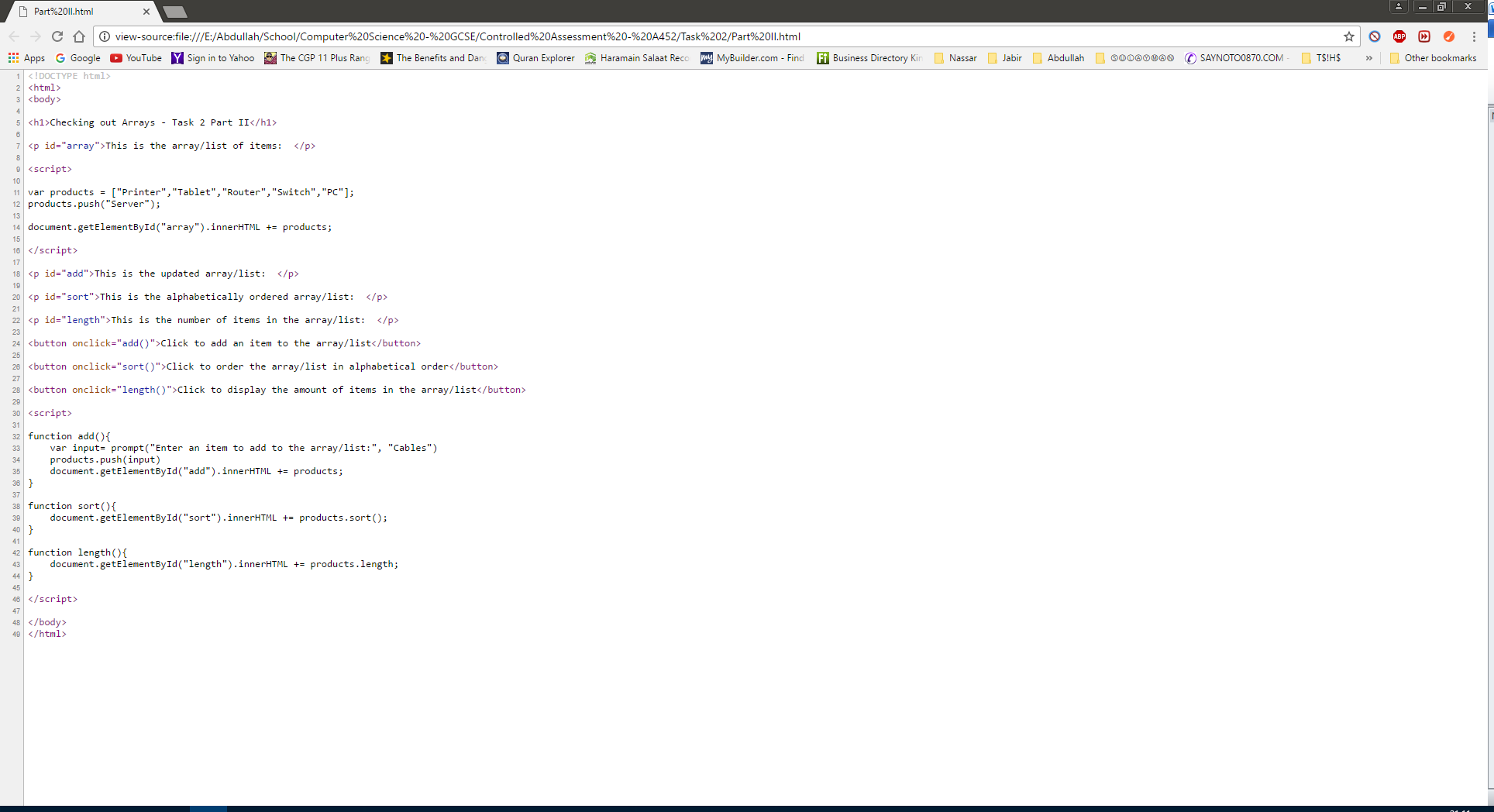
The script displays the introductory sentence only, when the button hasn’t been clicked.

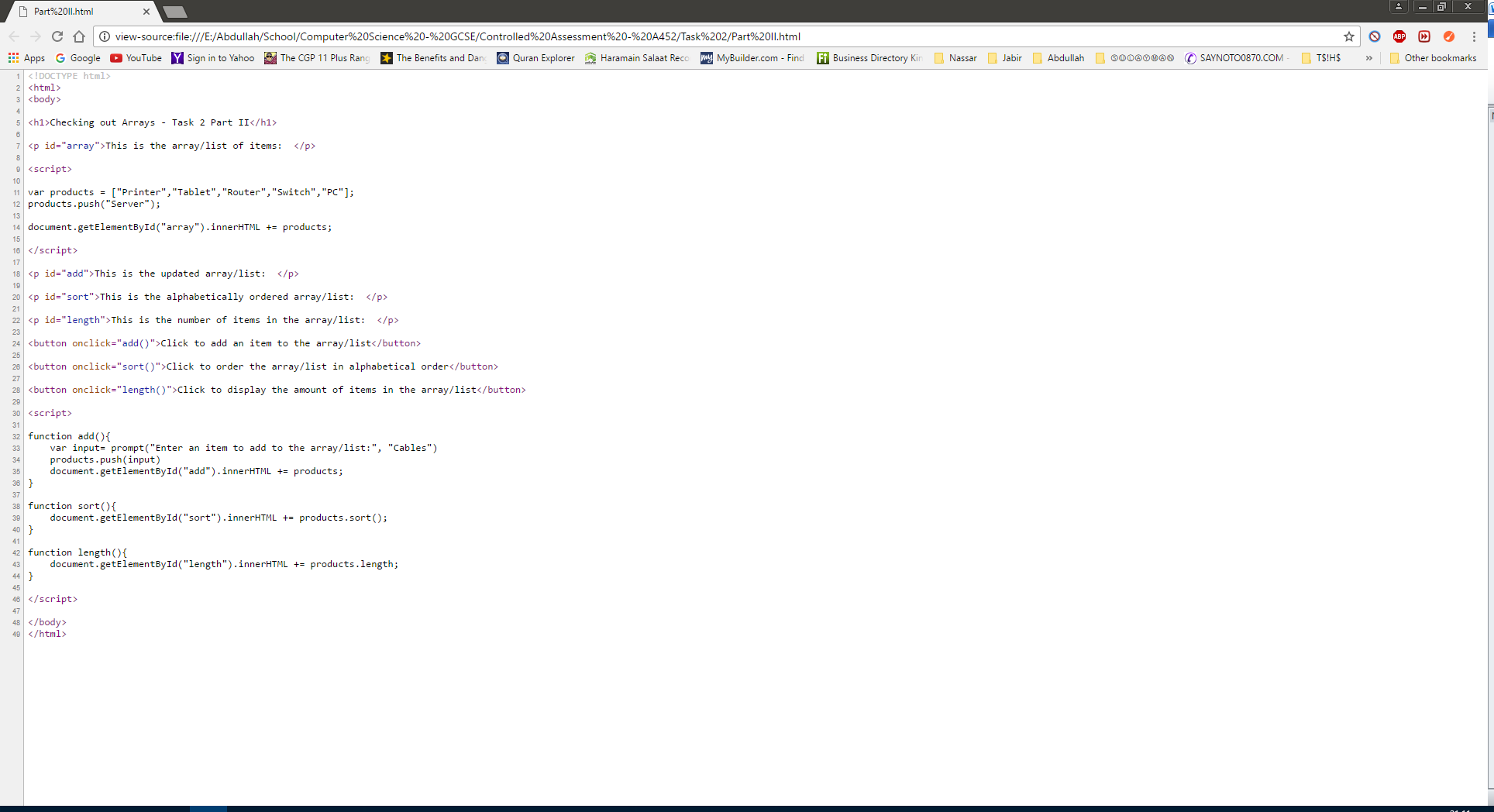


After the button was clicked, it displays the introductory sentence as well as the number of items in the list.

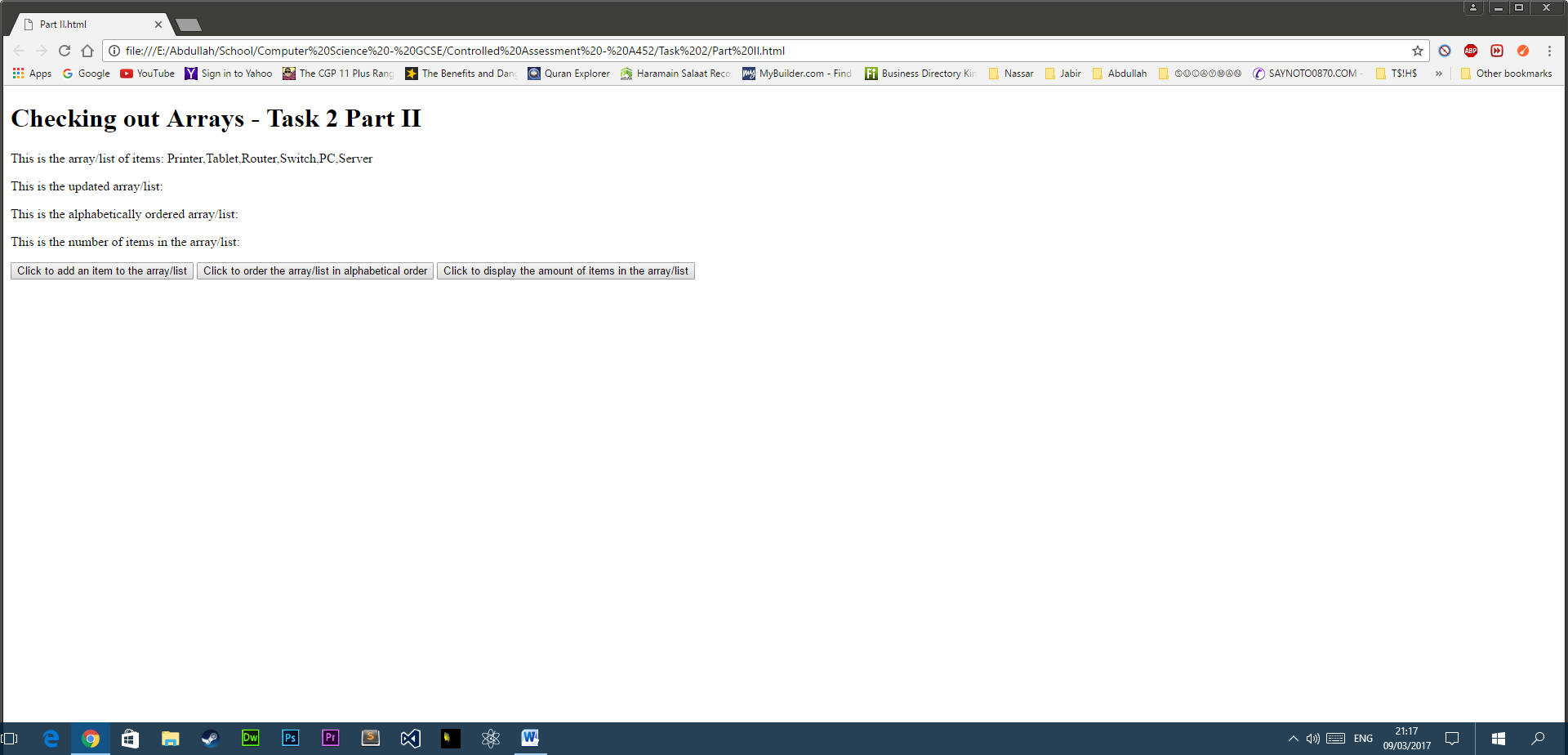
To further add functionality to the script, I added a feature which allowed the user to add an item to the array. This was done by first defining a new paragraph element with the id “add”, see **line 18**, to show the updated list as well as creating a button, see **line 24**, that invoked the function “add ()”, which adds an item to the array, by using the onclick event. The declared function uses the prompt method paired with a variable. The variable is defined and within it the prompt method contains a question to ask the user as well as providing an example of what the user can add, see **line 33**. The users’ input is then added to the array using the push () method, see **line 34**, and the updated list is then returned to the paragraph with the id “add” using the .innerHTML property, see **line 35**.







The script is successful as the item will be added and the new number of items in the list will be displayed, see images below.

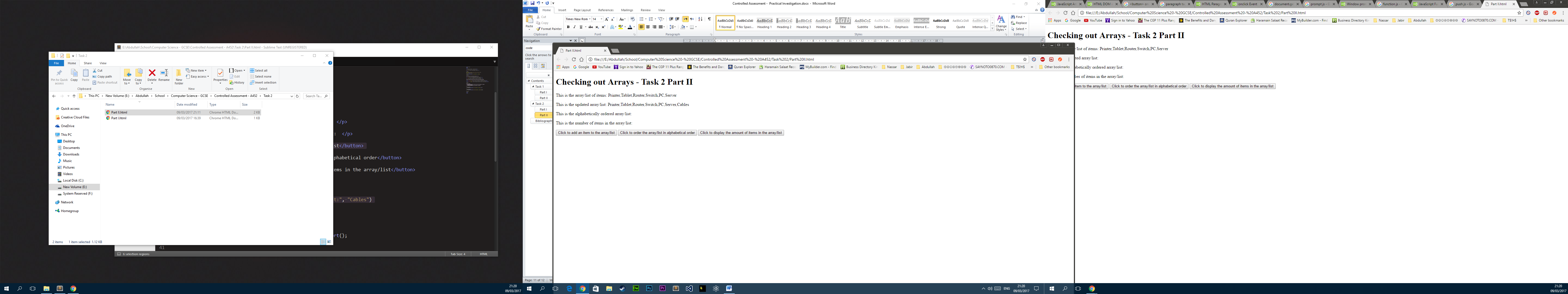


What the script displays when it is fully completed.

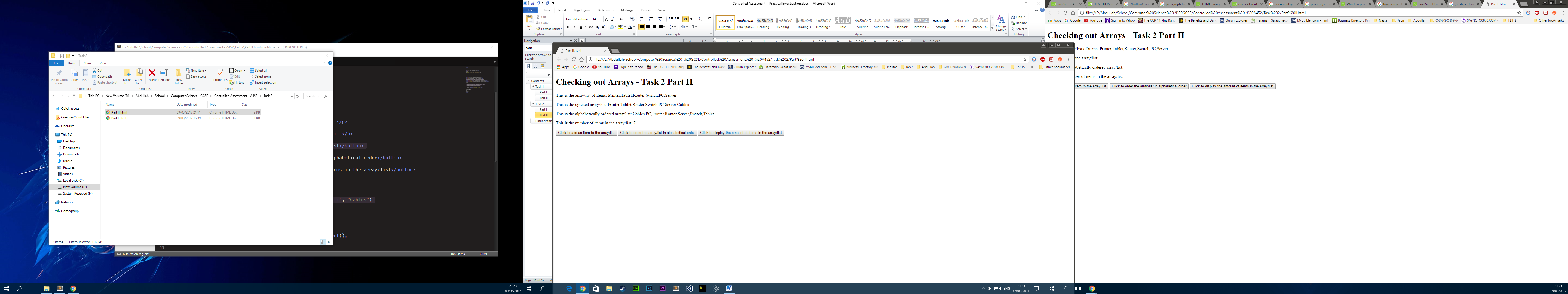
What the script displays when you click the button to add items to the list



What the script displays when you have added the item to the list.

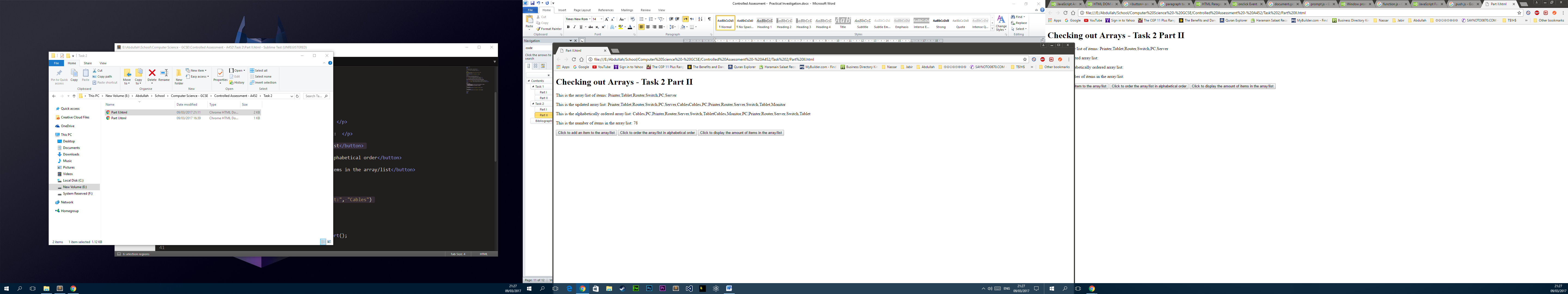


What the script displays when you have added the item to the list and would like to order it and see the number of items in the list.



Continued…

The only downfall, however, is that if you add more items, after you have added the first, they are correct but they are added onto the other lists and makes it look confusing.



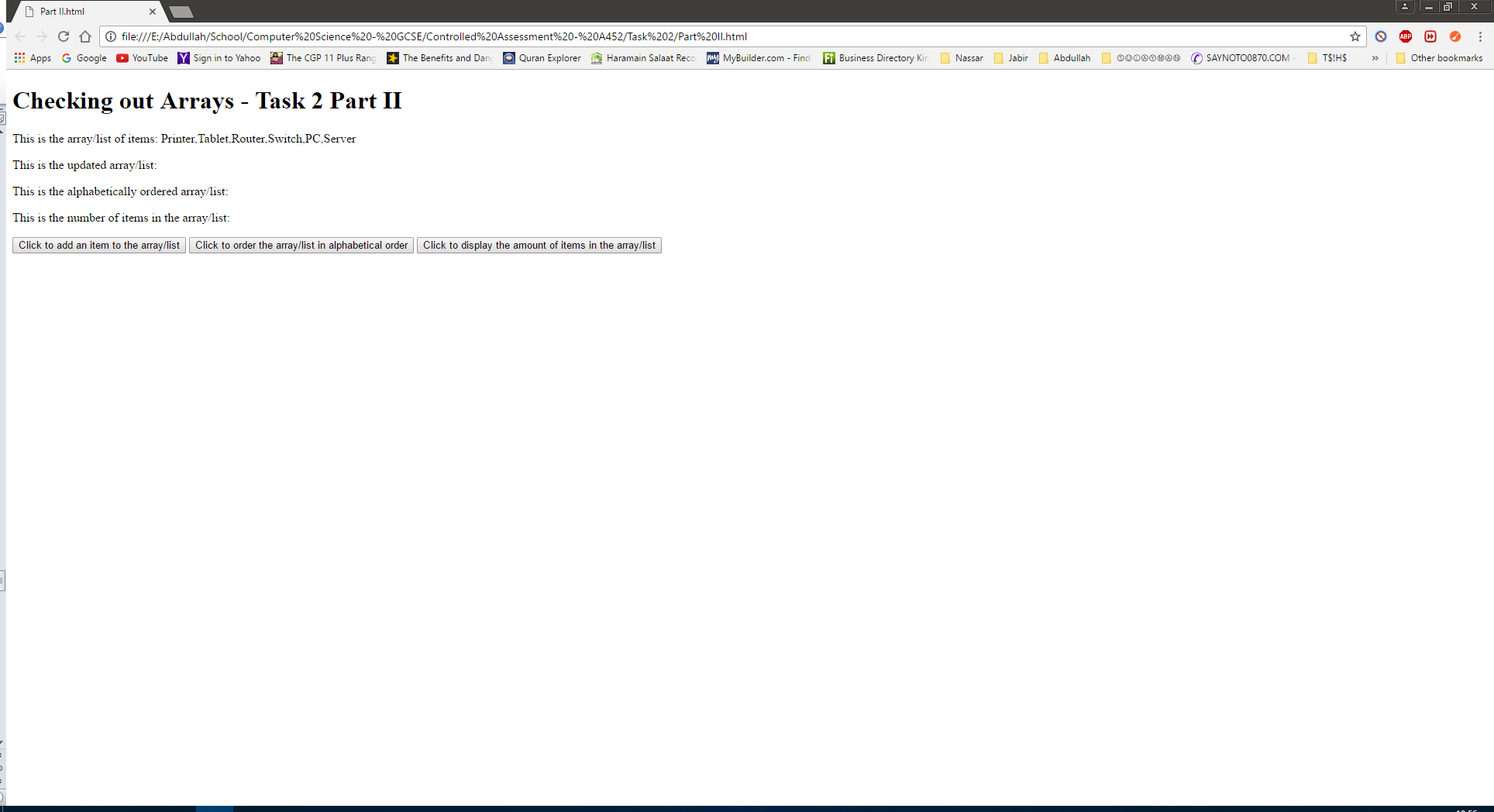
The items are added onto the original

#### Testing the code

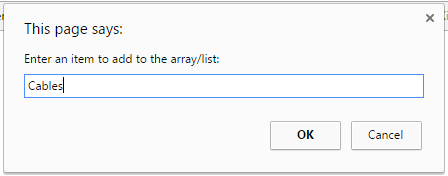
|  |  |  |
| --- | --- | --- |
| Criteria/Function tested: | Findings: | Success/Failure: |
| The array must be able to output the array alphabetically ordered. |  | Success |
| The array must be able to output the number of items in the array. |  | Success |

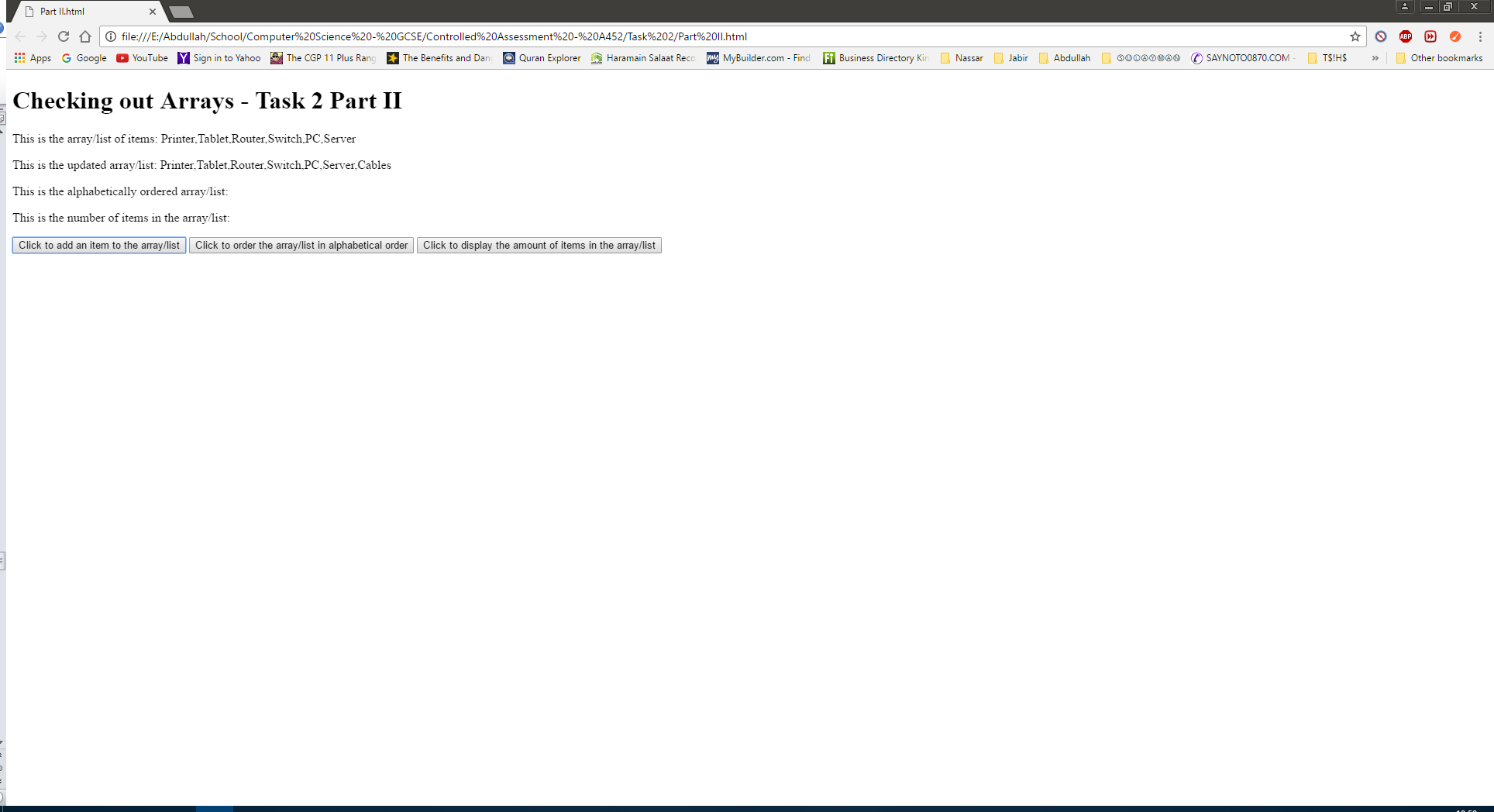
#### Demonstrating the code

When you open the webpage the user is greeted with a page containing a list of items, an updated list of items, including the user defined sentence, the alphabetically ordered list and the number of items in the list as well as three buttons allowing the user to add an item to the list, to order the list and to find out the number of items in the list.

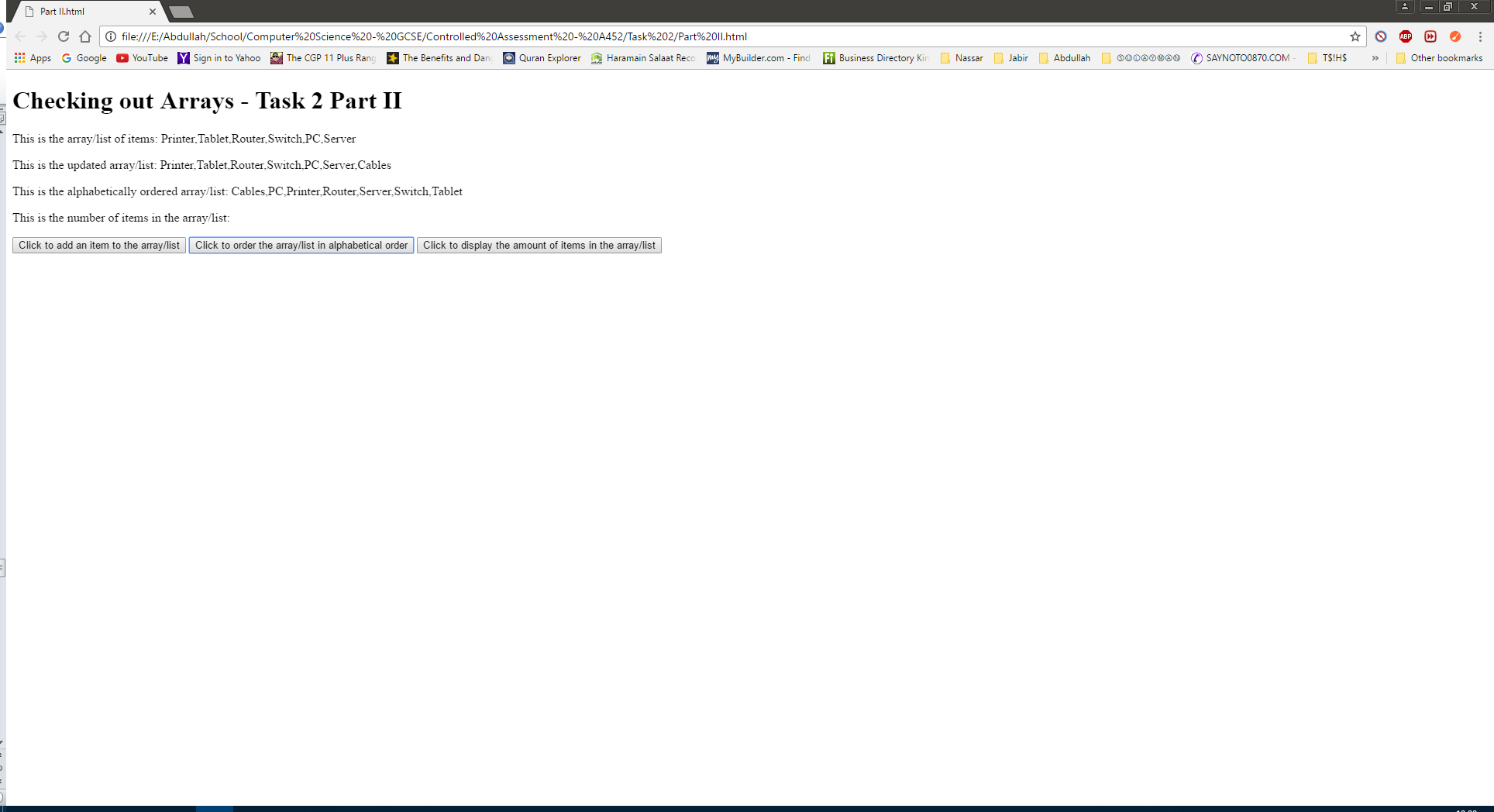


If the user clicks the “add an item to the list” button you are presented with a popup box to allow the user to write a word that will be added to the list which will be added, see image below.

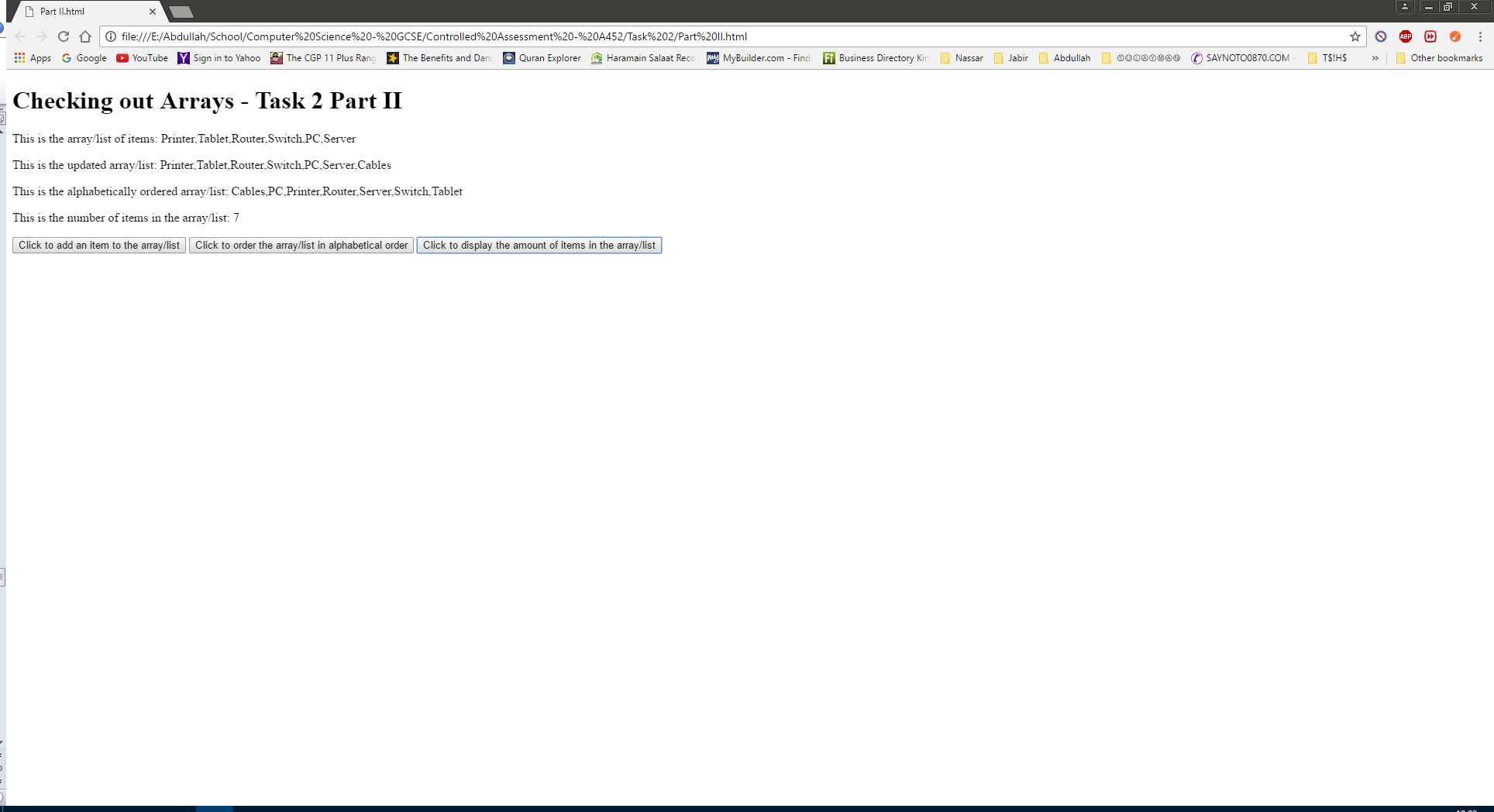




If the user clicks the “order the items in alphabetical order” button the webpage adds the ordered list to the web page, see image below.



If the user clicks the “display the amount number of items in the list” button the webpage adds the number to the webpage, see image below.



## Task 3

### Part I

The assets needed to make a traffic light include the images of the traffic lights at their different colour phases. We will need an image for red, red/amber and green. Furthermore, we will need a button so that the user can press and change the colours.

### Part II

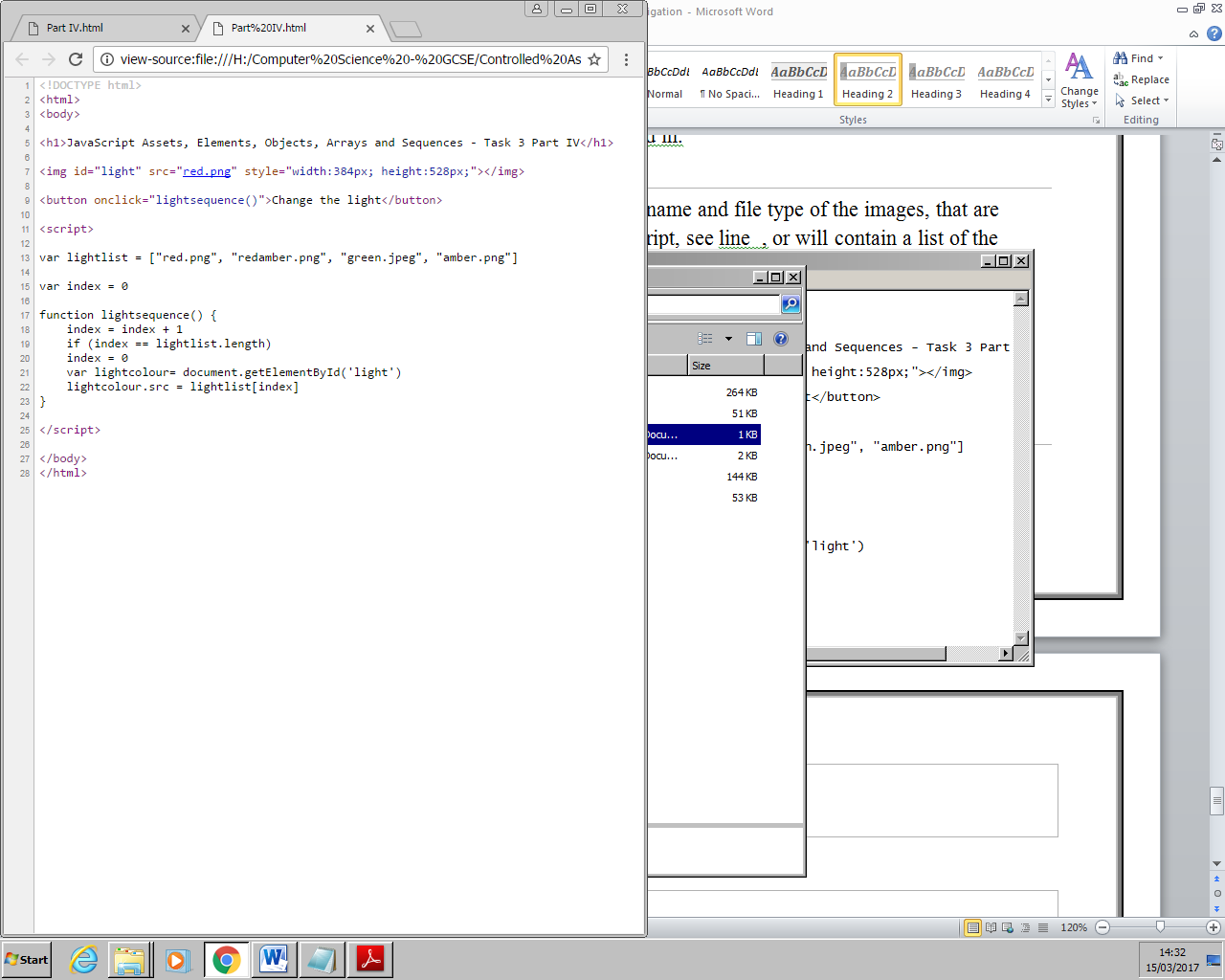
The assets will be, in this case, best located within the same folder that the script is in. These images will also be placed in an array in the source script so that they can be manipulated for the traffic light to work.

Alternatively, the images could be fetched by using an absolute URL that hosts the image, and effectively links the image to the HTML web page. This would be better because the user wouldn’t have to have the images downloaded on their PC, and would only need the HTML file which would be more user friendly. However, the problem is that the red/amber image doesn’t exist on the web and I had to edit an image to contain both colours which means that the image has to be located in the same folder as the script because it’s not online.

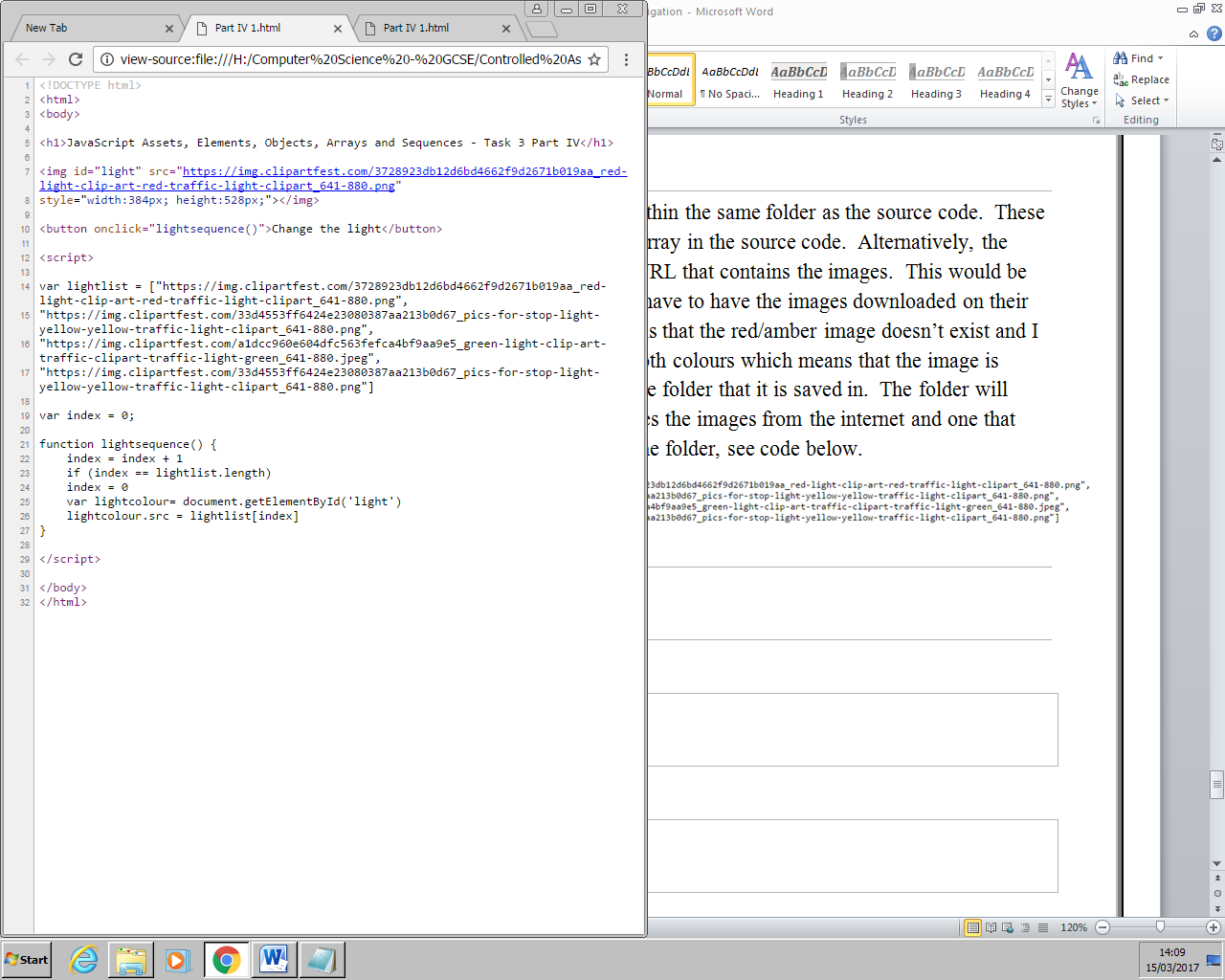
The folder, that the HTML file will be saved in, will contain two scripts. One that fetches the images from the internet, using the absolute URL method and using the red amber image from my folder, and one that simply fetches the images from the folder that contains the images, the same folder that the script is saved in.

### Part III

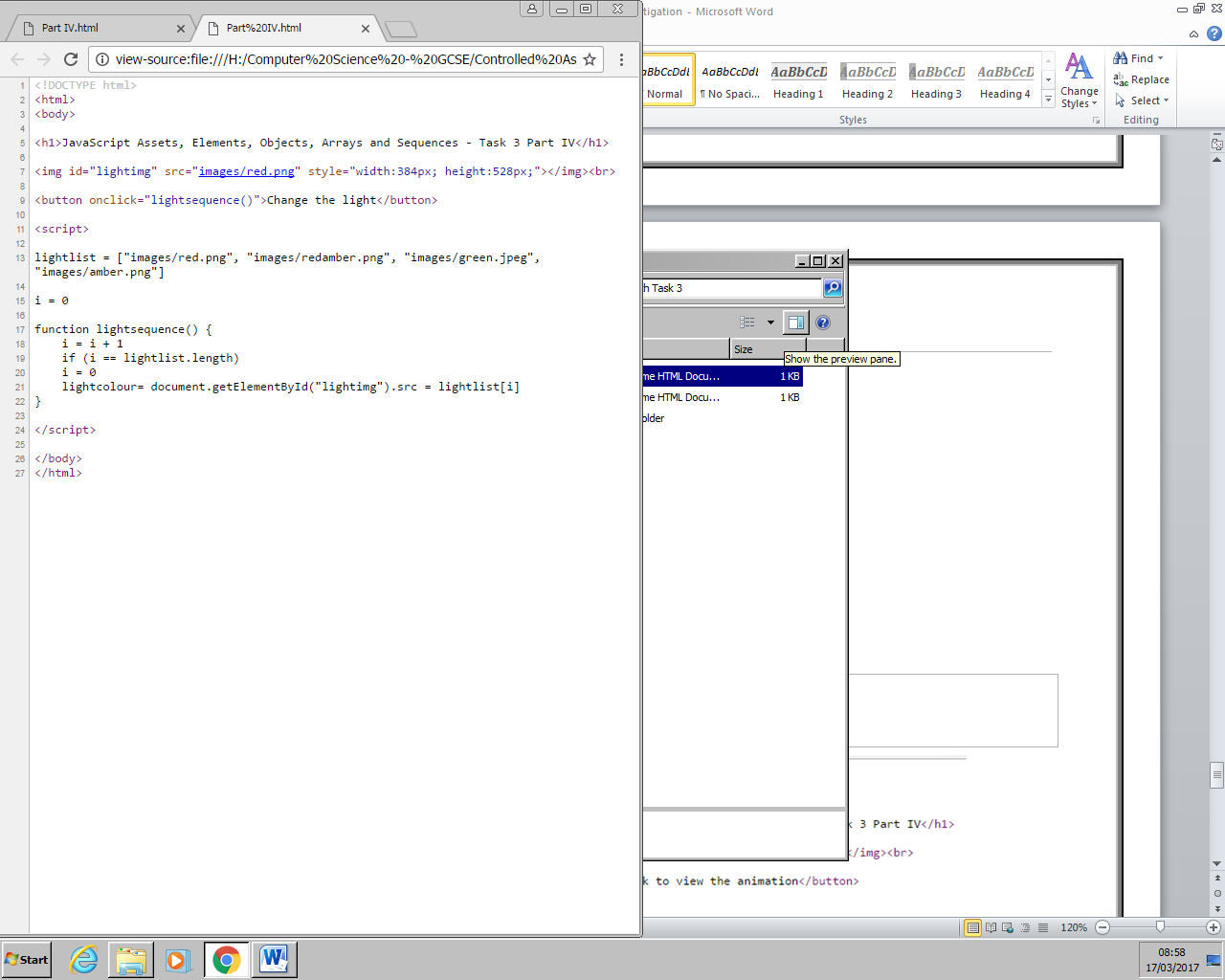
The array will either, contain the name and file type of the images, that are saved in the same folder as the script, see **line 13**, or will contain a list of the URL’s of the images, see **line 14-17**. For the reason why see **part II.**



Or



### Part IV



#### Code Explained

**Line 7** of the script defines an image tag and places an image, a traffic light, within it. This tag also defines the size of the image. This image is then manipulated by the script so that other images can replace it, so that the traffic light works.

**Line 9** simply creates a button that once pressed activates the function that changes the image of the traffic light.

**Line 13** is where an array is defined that contains a list of the images of the traffic light. The images are within a folder called images and so that needs to be mentioned within the array, so that the image can be found, and called, by the script.

**Line 15** defines a variable “i” which is used to index/number the traffic light images so that the traffic light sequence can work.

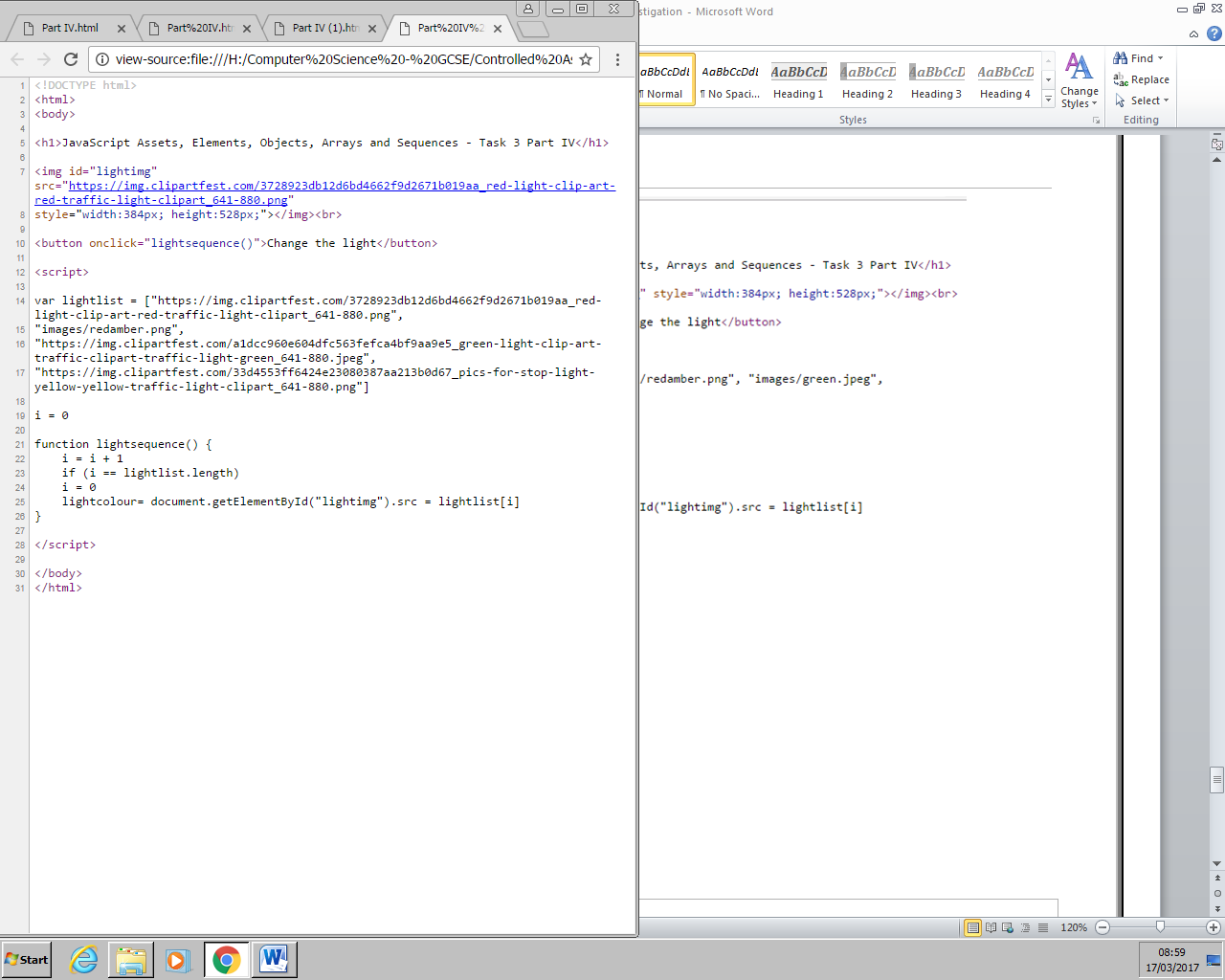
**Line 17** defines a function “lightsequence” that contains the script that changes the traffic light.

**Line 18** increments the value of the index by one, so that the next item in the array can be called.

**Line 19** contains an if statement that compares the index value with the length of the array and if they are equal to each other, the index value returns back to zero and the whole process starts again.

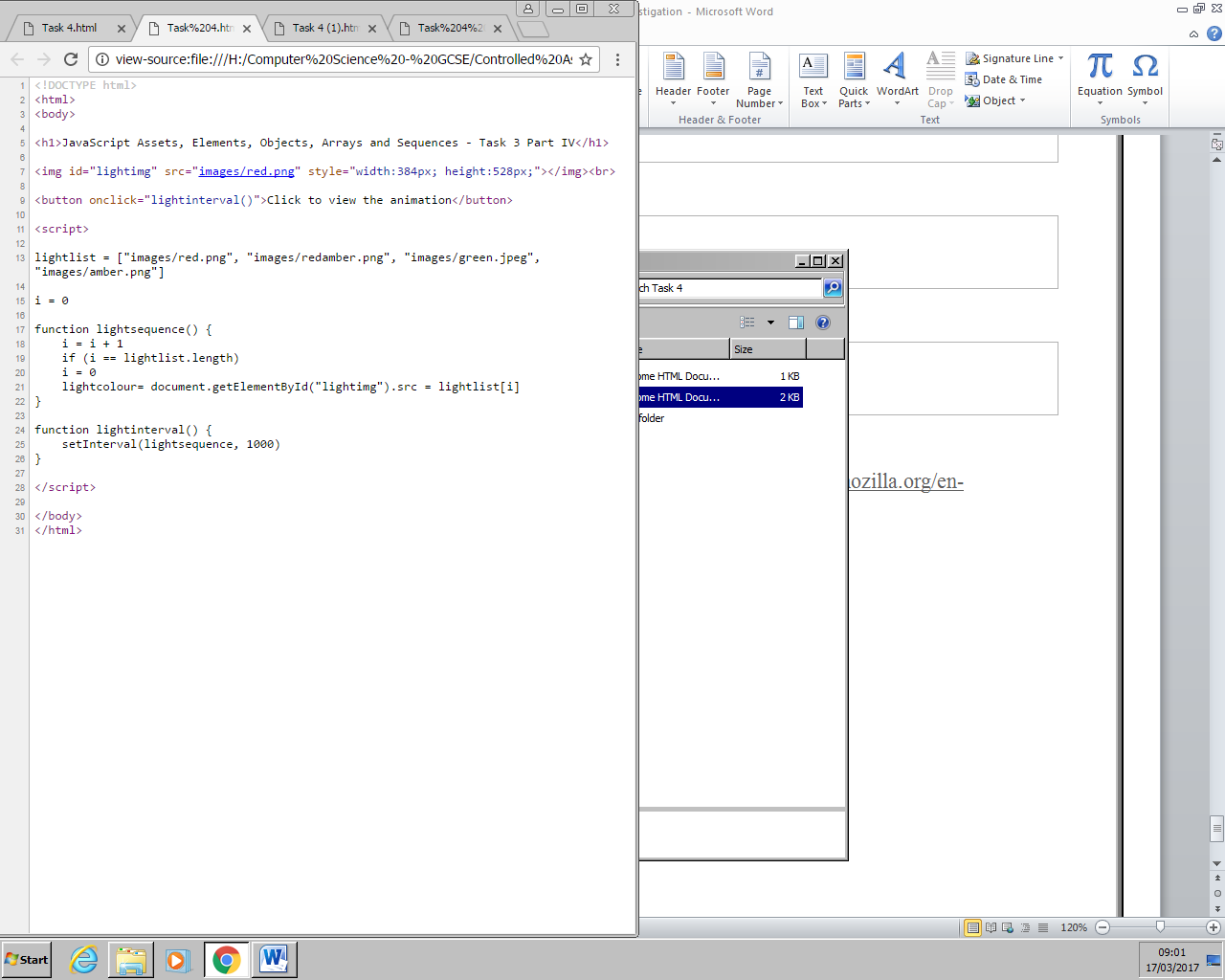
Regardless of the if statement, however, **line 21**, calls the image tag that was previously defined and then uses the .src property, which is similar to the .innerHTML property, except that it manipulates images rather than texts. The .src property is assigned the image that has the next index value in the list.

Or

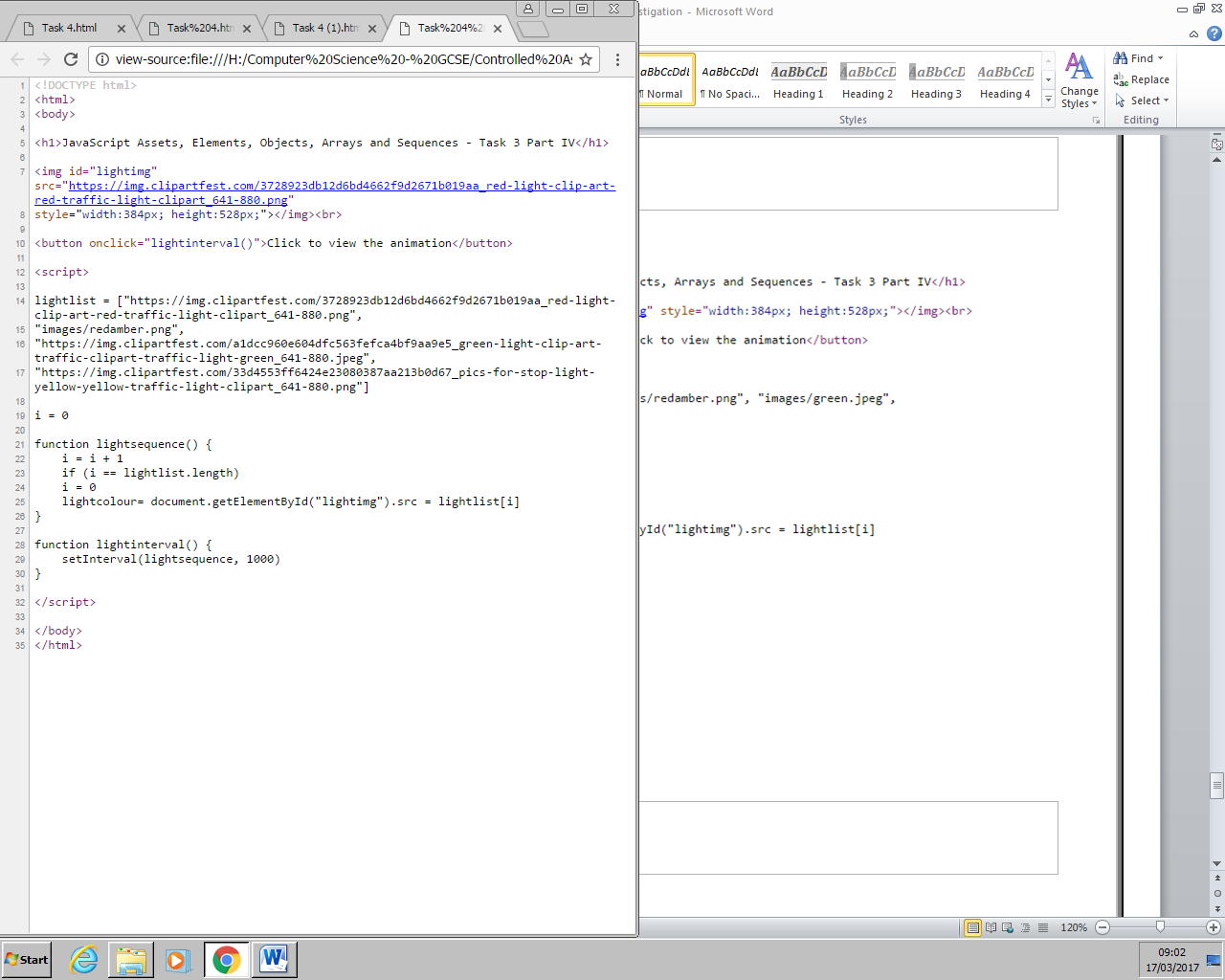


The only difference between this script and the other is the array that contains the images. In this case, line 7 and line 14-17, the image tag and the array contains an absolute URL that links the image, that’s on the internet to this script.

## Task 4



Or



## Task 5

### Benefits of saving JavaScript scripts in the HTML of webpages

One of the main benefits of placing the script within the HTML is that these webpages will load faster and more efficiently. This is because the web browser needs to reference an external code and thus make a separate request for the web server to fetch the external JS. This ultimately means that the webpage loads faster because it doesn’t need to send the request and doesn’t need to fetch the code but only needs to continue to read the HTML code. This is also efficient because it means fewer requests are needed and less bandwidth is used up, which is especially useful for mobile users who have a strict mobile data plan.

### Drawbacks of saving JavaScript scripts in the HTML of webpages

One of the main drawbacks is that the script takes up line space in the source code. This has many ramifications, the main one being that the script becomes less readable. If the script becomes less readable, it means bug finding and noticing mistakes is a lot harder as well as making the files larger which, in turn, makes maintaining them harder.

### Benefits of saving JavaScript scripts externally as script files

One of the benefits of saving the scripts externally is that it allows for caching. By making the JS files cached it speeds up page loading times. This is because the browser simply uses the cached script instead of requesting a new one which in turn allows for quicker load times. Furthermore, if the user moves to another webpage that contains the same external JS reference, the users browser would have cached the script and so the browser wouldn’t have to request the script and thus loading faster.

Another benefit is the fact that its easier to maintain both HTML and JavaScript code/script. This is because the scripts are more readable because essentially the files are smaller and easier to work with. Furthermore, the fact that its stored externally means that if the script is needed on multiple webpages, both HTML codes of the webpages can simply reference it rather than writing the whole code on both and thus making the files larger. This also means that if a change needs to be made, it can be made to one file rather than accessing all of the webpage HTML codes and changing the script. Finally, the fact that you have fewer lines of HTML code means that search engine spiders can more thoroughly iundex your site and find the important keywords which makes your webpage more accessible by the users.

One of the main benefits with regards to the maintenance of webpages is that external JavaScript files work well with version control systems such as GIT or SVN. This means developers can easily modify webpages without any complications. Furthermore, these smaller JS files can be minified, removing unnecessary characters without changing its functionality which can be done using Googles closure compiler or the YUI compressor.

### Drawbacks of saving JavaScript scripts externally as script files

One of the main problems is that of dependency. This is when one script relies on another script and for some reason the script fails to download, the second script won’t be downloaded and the whole application will crash. This also leads onto the fact that if many webpages use a common JS file, the webpages depend on the script and if you change the common JS, some of the webpages may not comply with the script.

One of the drawbacks of saving JS files externally is that these files can easily be downloaded by using the URL of the JS file. This ultimately means that the code can easily be stole.

## Bibliography

W3School: <http://www.w3schools.com/js/default.asp>

The Executionists: <https://www.executionists.com/>

Mozilla Developer Network (MDN): <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Knowledge Hills: <http://knowledgehills.com/javascript/javascript-tutorial.htm>

Tizag Tutorials: <http://www.tizag.com/>

Stack Overflow: <http://stackoverflow.com/>

Computer Hope: <http://www.computerhope.com/>

Guru 99: <http://www.guru99.com/interactive-javascript-tutorials.html>

TechOnTheNet: <https://www.techonthenet.com/>

Software Engineering Stack Overflow: <http://softwareengineering.stackexchange.com/>

Wikipedia: <https://en.wikipedia.org/wiki/Main_Page>  
Techwalla: <https://www.techwalla.com/>