COMP1562 – Operating System

Laboratory 5

Shell Programming 2

Group ID: 21

Group Task: Task 4

Usman Basharat

Mohamed Aden

Yunus Hassan

Derek U. Mafohla

Table of Contents

[Task 4.1 3](#_Toc475221176)

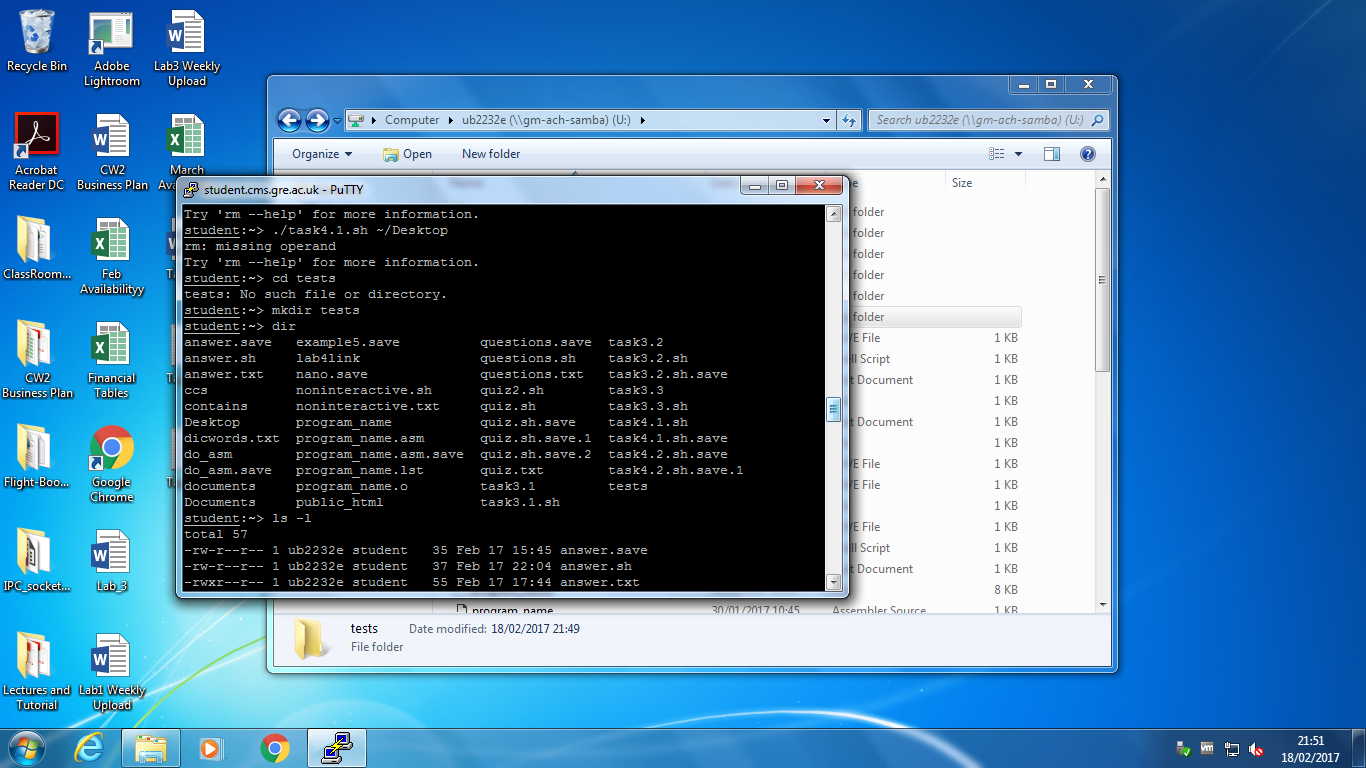
[Task 4.2 4](#_Toc475221177)

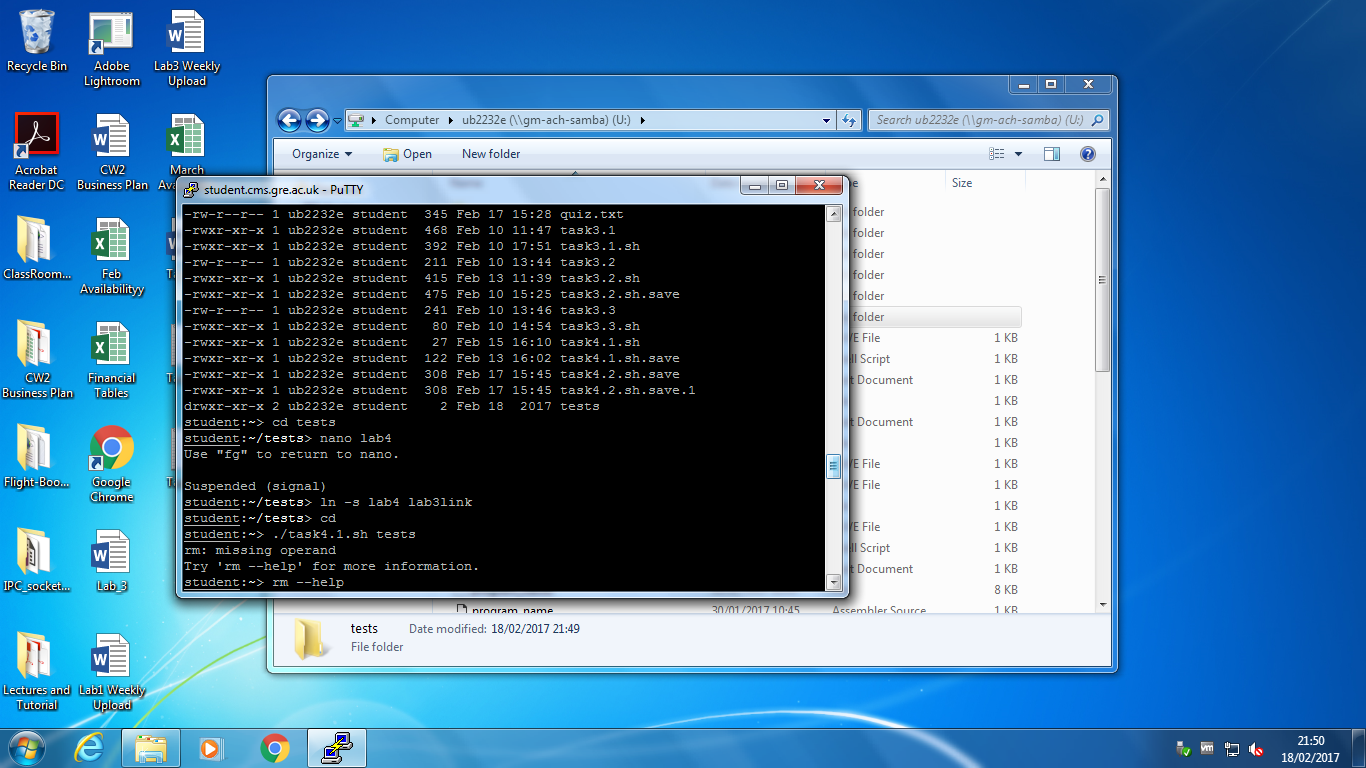
[Reflection 6](#_Toc475221178)

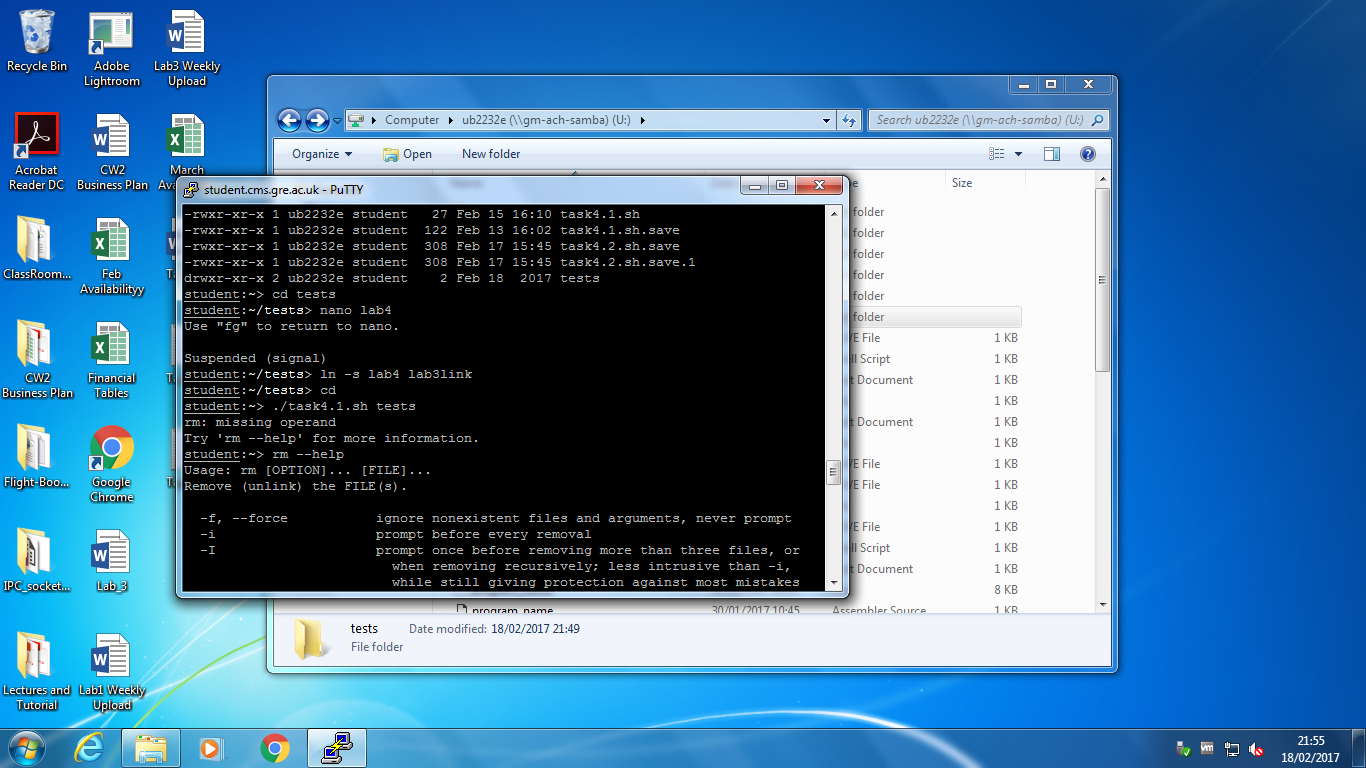
# Task 4.1



Figure 1 shows the code for Task 4.1

Figure 4.1 shows the code to execute broken symbolic links. As soon as the piping method is used to search for the broken links within the directory, it removes it. As I was doing this, I made a mistake of searching for symbolic links, and removing them. After reading the laboratory document, I realised it is the complete opposite. Therefore, when I ran the line of code through script check, it worked and I moved on to the next task. The screenshots below are the following of how this lab has been executed. The first movements were to create a directory and make an empty file. Once this was created, we need to create the link. Once the link was created, we make it broken by removing it and then we tested our code to remove it and it worked shown on Figure 6

Figure 2 shows making a new directory

 Figure 3 shows creating a new file

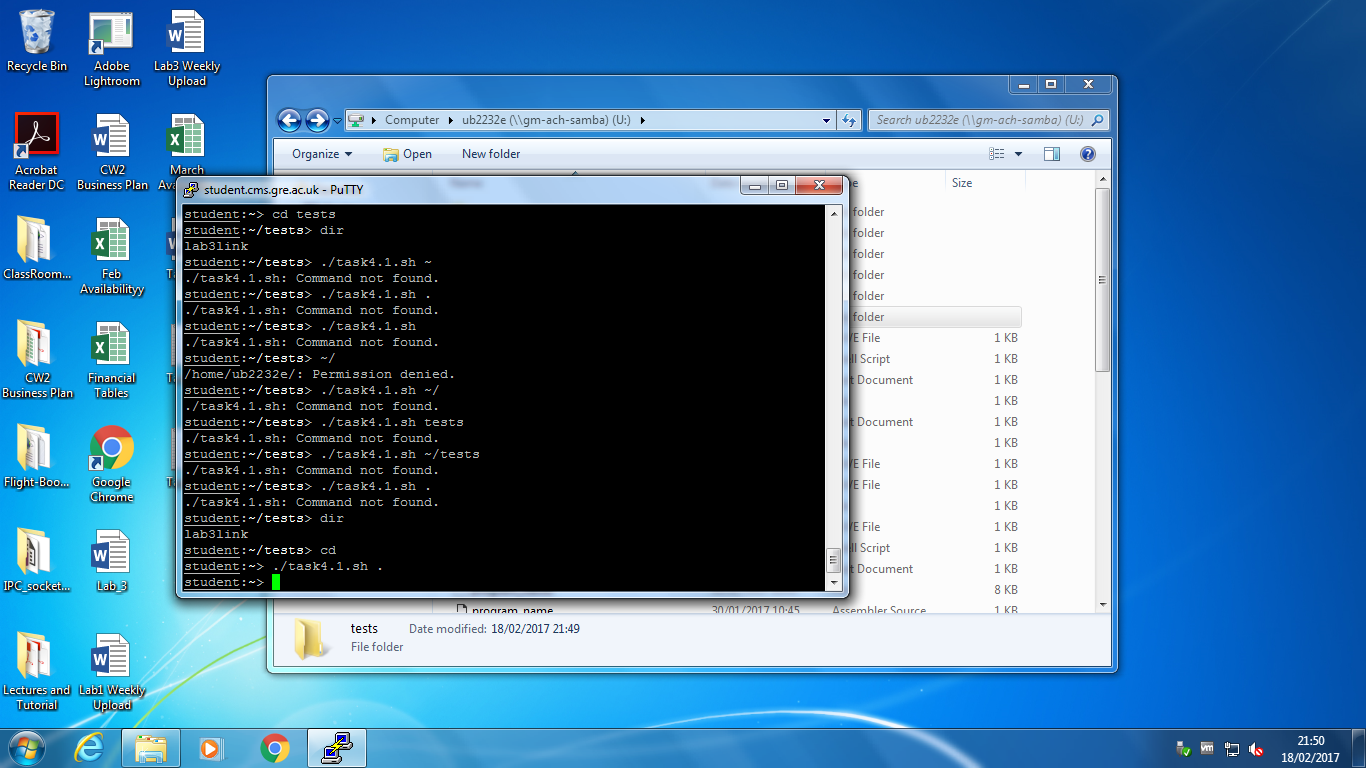
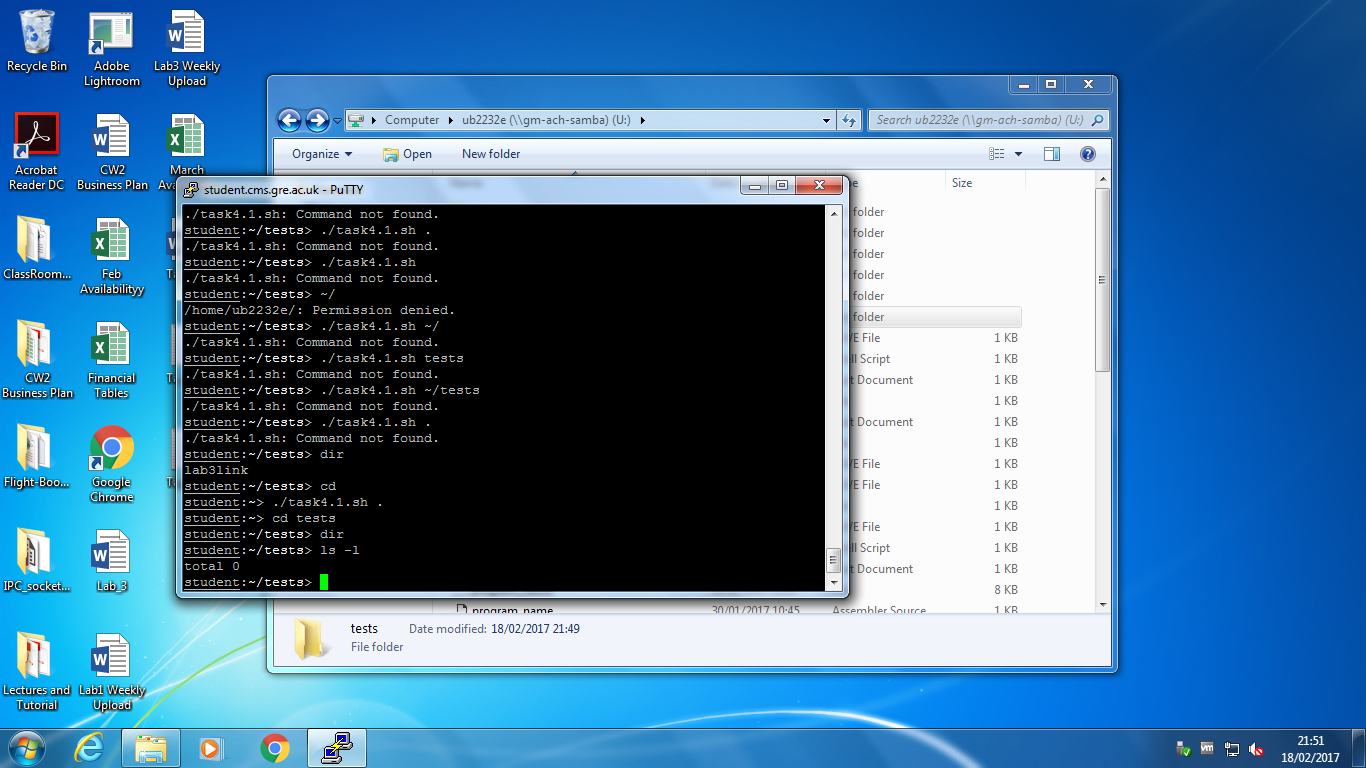
Figure 4 shows creating a symbolic link between the new file created

Figure 5 shows the broken link once the new file has been created

Figure 6 shows the symbolic link removed

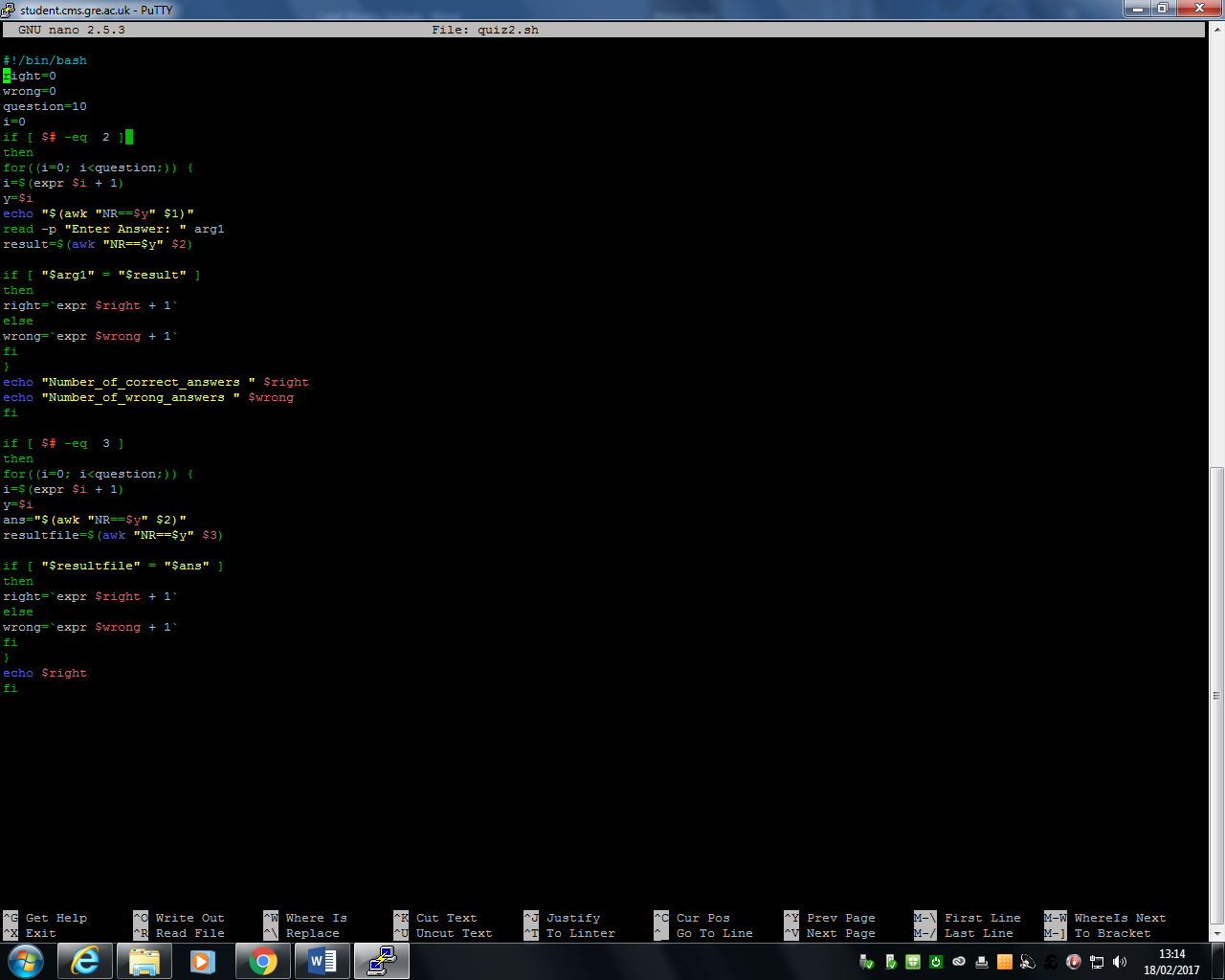
# Task 4.2

The code is split between in two. This works by the first section highlighted in blue is recognised by the interactive mode of the quiz. This means that the user has to type in the answer they feel and they would receive the result. Once the user has typed in the result, this is matched by the correct answers that has been typed in. Each answer that is wrong, $wrong is added one and the same with $right.

However, the other section highlighted in green is the non-interactive part. This part shows that there are non-interactive answers that have been already typed in shown in Figure 9 and is compared to the actual answers shown in Figure 8. This only displays the correct answer. Once it executed, it matches the two and it displays how many answers are correct.

How the code is run for both is similar. The first loop runs 10 times, matching how many questions there are. Each time it runs, it uses the arguments to display the question and to match the answer. This is a clever loop, because once it is executed, it matches the arguments and selects which one is needed to execute.

Figure 10 and Figure 11 shows the difference between the two codes.



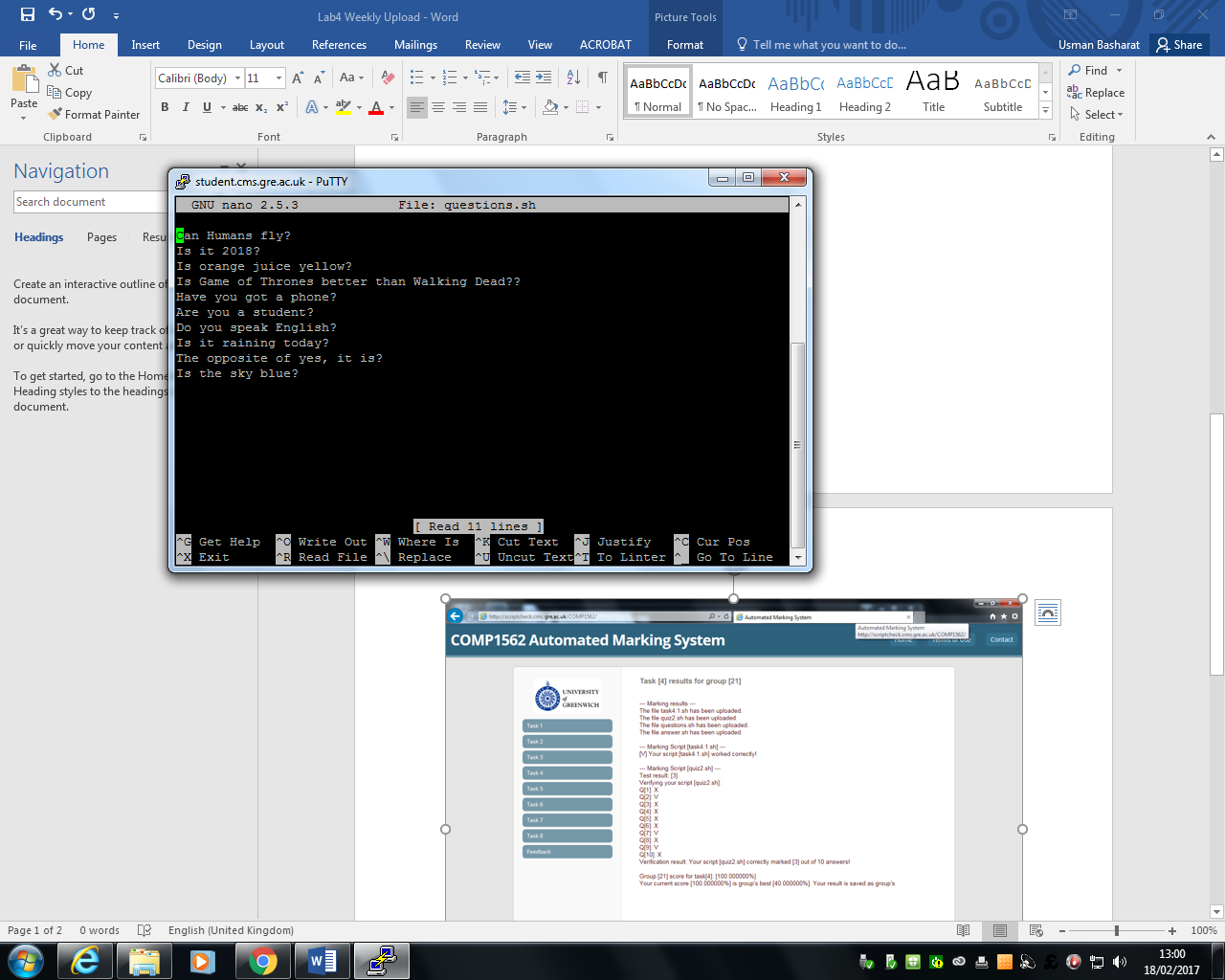


Figure 7 shows the questions used for this task

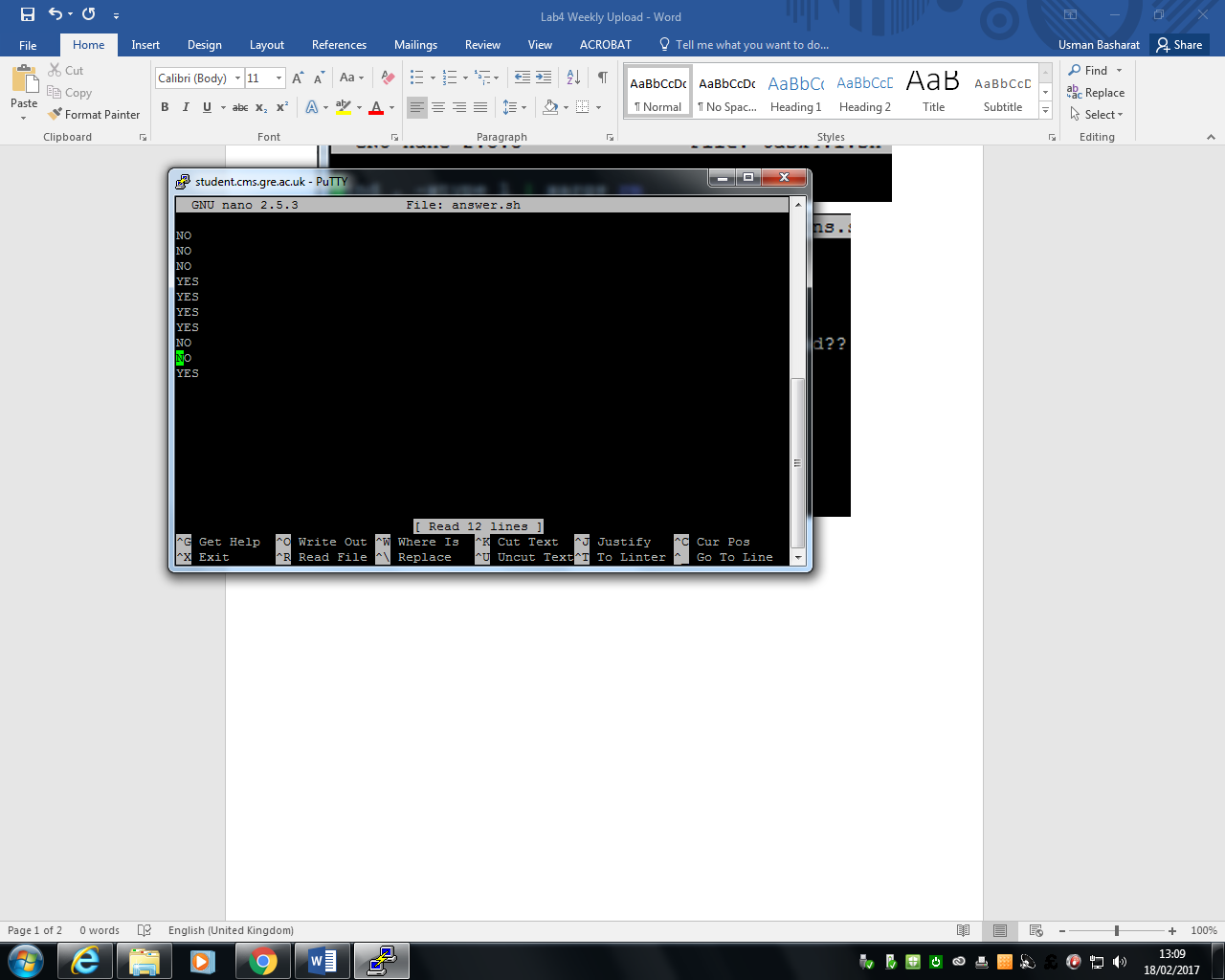


Figure 8 shows the answers to the questions for this task

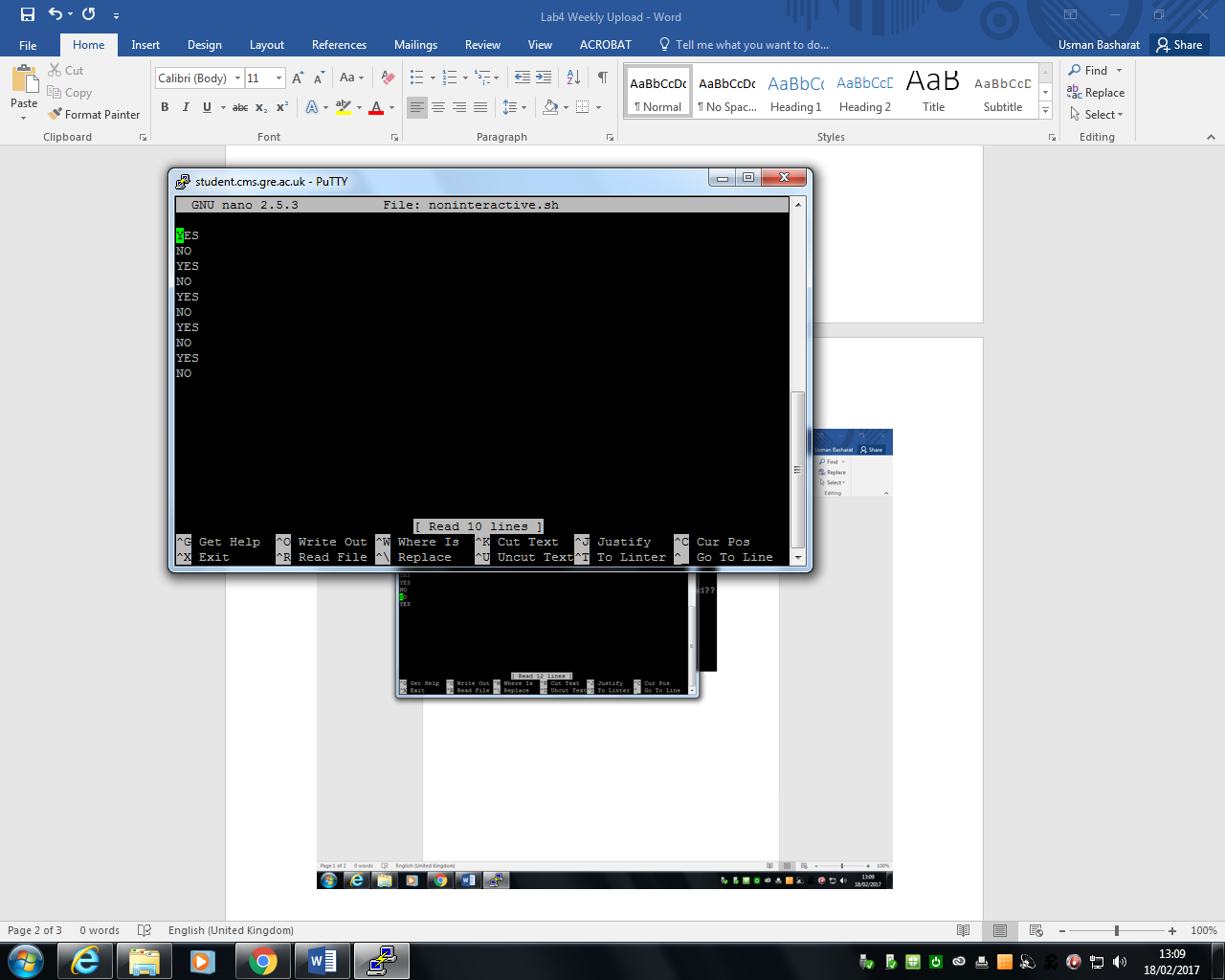


Figure 9 shows the non-interactive questions used.

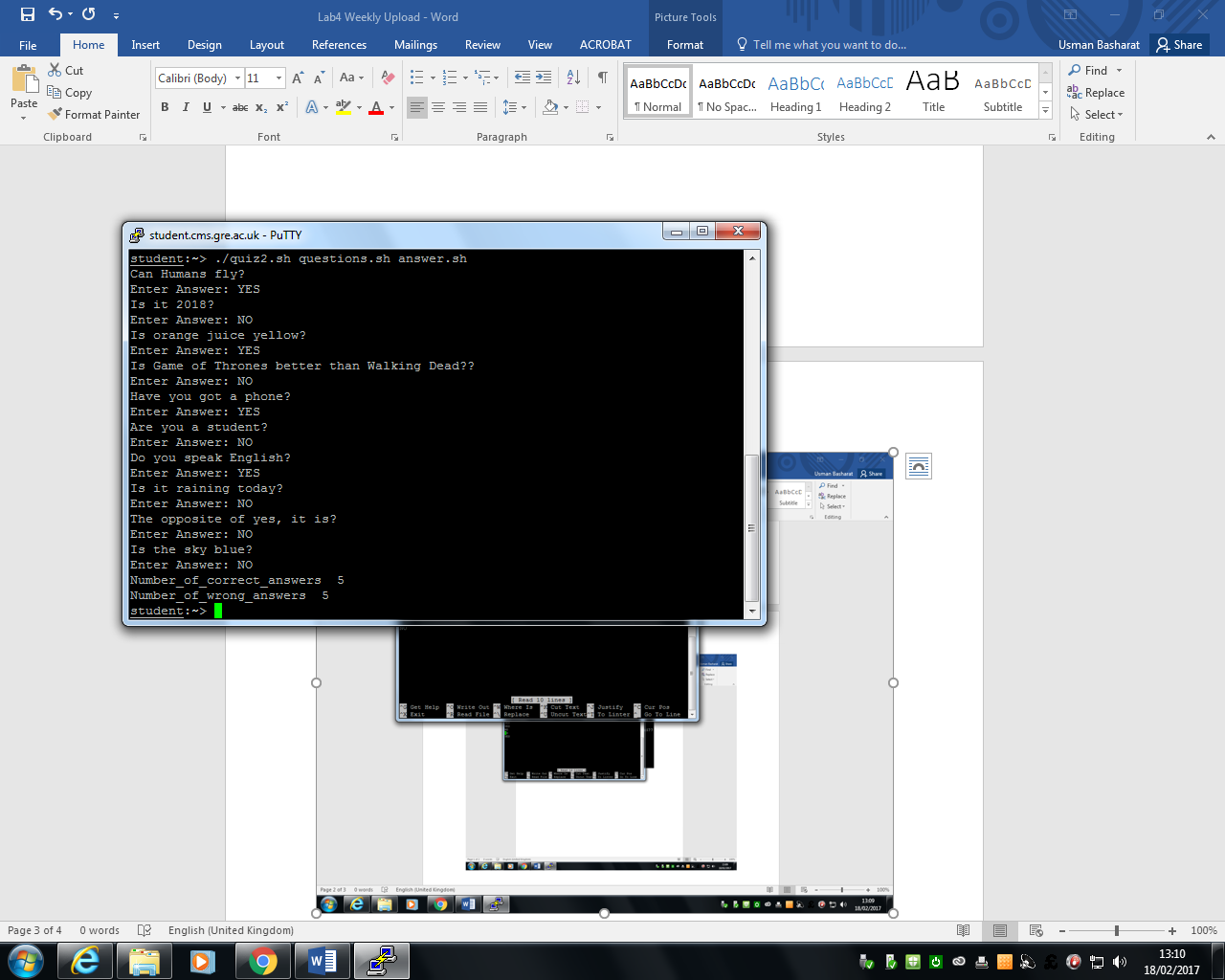


Figure 10 shows the interactive quiz.

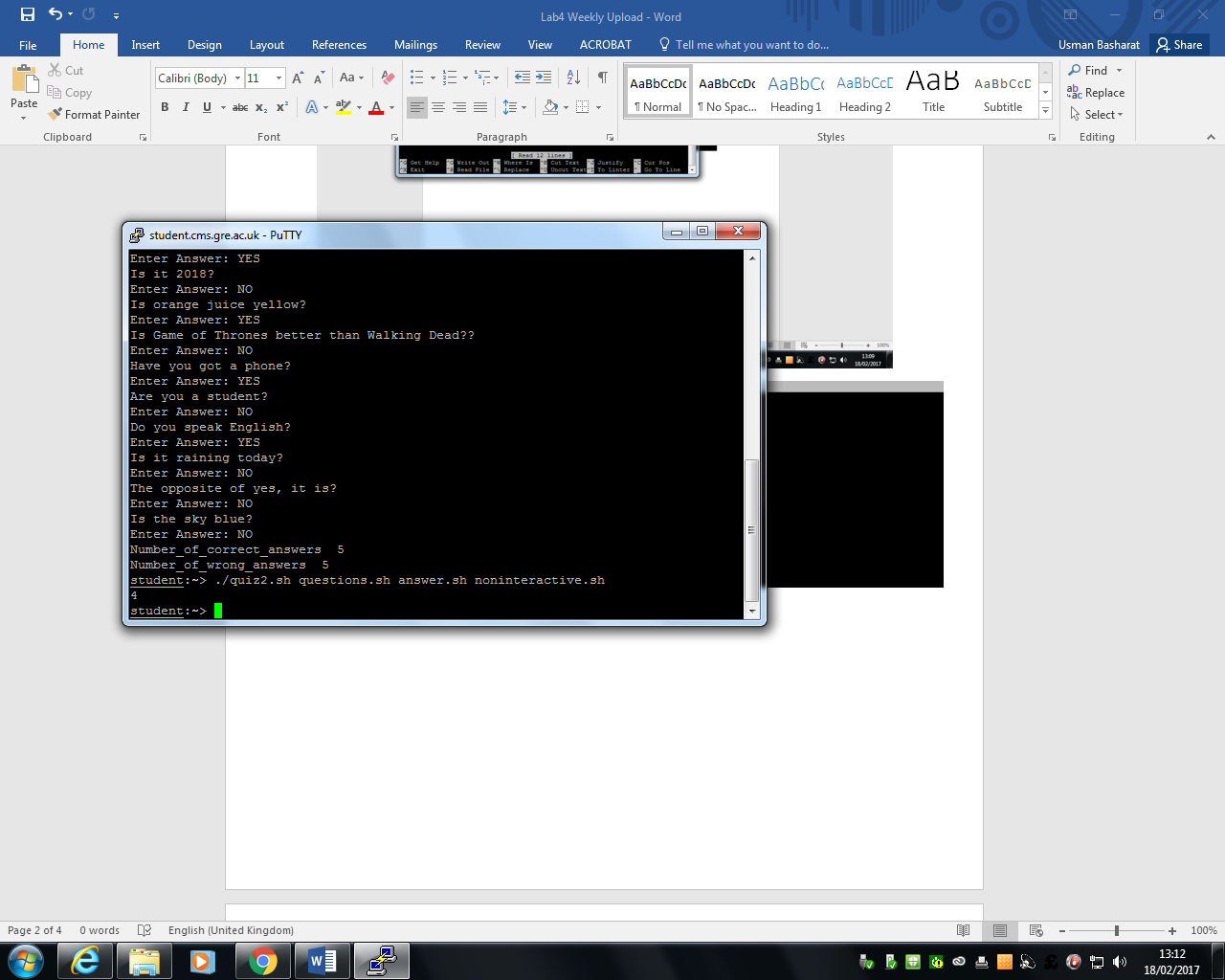
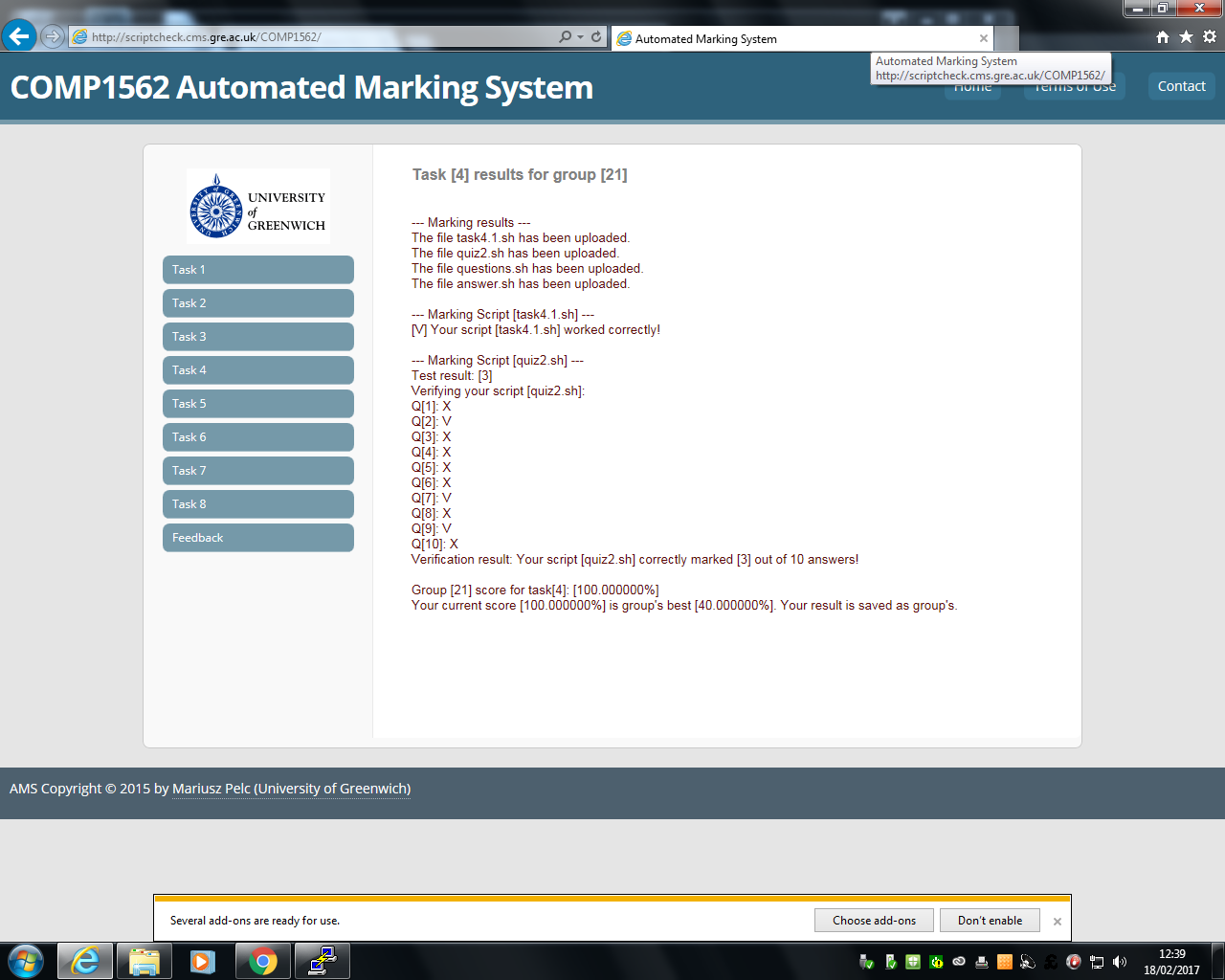


Figure 11 shows the non-interactive quiz



# Reflection

During the time for this laboratory, I found this task tough again. I felt that the first one was not as tough as the second one, because we had to link three files together. It was hard to put the non-interactive part of the quiz in, but when I understood the first part of how I did it, it was easier to understand the non-interactive part. I felt that I spent much time on this the most, because it was the most challenging one out of them all. These challenges took time and effort to be completed and I felt that with my group members, we were able to complete this task and get the result above. Overall, I felt that I have understood the basic understanding and importance of how to run and produce the code effectively.