Q1:

(Sections 4.5 - 4.7) Consider the following code segment.

or $t2, $t0, $t1  
 or $t3, $t1, $t2  
 or $t4, $t2, $t3

We want to execute this stream of instructions on a 5-stage pipelined MIPS processor.

Assume the processor does not implement forwarding.  Draw a stage diagram similar to Figure 4.34 on p. 288 showing the execution of each stage of the three instructions, assuming the processor fetches the next instruction each clock cycle.  Identify all the data hazards.  Upload a picture of your drawing as a PNG or PDF file.

Q2:

(Sections 4.5 - 4.7) Consider the following code segment.

or $t2, $t0, $t1  
 or $t3, $t1, $t2  
 or $t4, $t2, $t3

We want to execute this stream of instructions on a 5-stage pipelined MIPS processor.

Assume the processor does **not** implement forwarding.  Draw a stage diagram, only this time insert **nop**s between regular instructions, as necessary, in order to stall the pipeline long enough to ensure that the instructions execute correctly.  Upload a picture of your drawing as a PNG or PDF file.

Q3:

(Sections 4.5 - 4.7) Consider the following code segment.

or $t2, $t0, $t1  
 or $t3, $t1, $t2  
 or $t4, $t2, $t3

We want to execute this stream of instructions on a 5-stage pipelined MIPS processor.

In ALU-to-ALU forwarding, the output of the ALU for a particular instruction is available as an input of the ALU of the following instruction at the beginning of the next clock cycle. Draw another diagram like you did in question 7b showing the execution of each stage of the pipeline, this time assuming that the processor uses ALU-to-ALU forwarding. Show the forwarding by drawing a line from the ALU output of one instruction to the ALU input of the instruction which needs it.  Upload a picture of your drawing as a PNG or PDF file.

Q4:

(Section 4.8) Assume a particular branch in code is encountered several times and the taken (T) / not-taken (NT) behavior at that branch turns out to be: **T, NT, T, T, NT**

What is the accuracy of the **always-taken** predictor for this sequence of branch outcomes.  Enter your answer as a percent value between 0 and 100.

Q5:

(Section 4.8) Assume a particular branch in code is encountered several times and the taken (T) / not-taken (NT) behavior at that branch turns out to be: **T, NT, T, T, NT**

What is the accuracy of the **never-taken** predictor for this sequence of branch outcomes.  Enter your answer as a percent value between 0 and 100.

Q6:

(Section 4.8) Assume a particular branch in code is encountered several times and the taken (T) / not-taken (NT) behavior at that branch turns out to be: **T, NT, T, T, NT**

Assume we use a **1-bit predictor**, where 0 means the previous branch was not taken and 1 means it was.  Assuming the state is initialized to 0, what is the accuracy of the of a 1-bit predictor for this sequence of branch outcomes?  Enter your answer as a percent value between 0 and 100.

Q7:

(Section 4.8) Assume a particular branch in code is encountered several times and the taken (T) / not-taken (NT) behavior at that branch turns out to be: **T, NT, T, T, NT**

Assume we use a **1-bit predictor** initialized to 0, but this block of code is revisited 10 times, so the sequence at the branch become {**T, NT, T, T, NT} x 10** (for a total of 50 branches).

What is the accuracy now? Enter your answer as a percent value between 0 and 100.

Q8:

(Section 4.8) Assume a particular branch in code is encountered several times and the taken (T) / not-taken (NT) behavior at that branch turns out to be: **T, NT, T, T, NT**

Assume we use a **2-bit predictor**, where 00 means the previous branch was not taken twice (or more) and 11 means it was taken twice (or more).  Assuming the state is initialized to 00, what is the accuracy of the of a 2-bit predictor for this sequence of branch outcomes?

Q9:

Section 4.8) Assume a particular branch in code is encountered several times and the taken (T) / not-taken (NT) behavior at that branch turns out to be: **T, NT, T, T, NT**

Assume we use a **2-bit predictor** initialized to 00, but this block of code is revisited 10 times, so the sequence at the branch becomes **{ T, NT, T, T, NT } x 10** (for a total of 50 branches).

What is the accuracy now? Enter your answer as a percent value between 0 and 100.