Question 1: (project1)

We input two binary values A and B into our And4 chip which performs a bit-wise ‘and’ for two 4-bit inputs. A = 0101 and B = 0011. What is our output?

1. 0111
2. 1000
3. 🡪0001
4. 0110

Question 2: (Lecture2)

In this lecture we are told that the ALU must find ‘zr’ and ‘ng’ chips. What do these chips represent, and how do they benefit us when we later build the CPU?

1. ‘zr’ returns 1, indicating an error, if I’m calculating something illegal in my ALU, (for example the square root of a negative number), ‘ng’ lets me know if the output was negative. This helps me avoid computer crashes by possibly outputting imaginary values.
2. ‘zr’ counts the number of zeros in the output of the 16 bit bus. ‘ng’ tells me if both input strings where negative or not. These help me iterate through the 16 bit buses of the input and output effectively, and reduce runtime
3. 🡪‘zr’ lets me know if my output equals zero, while ‘ng’ lets me know if my output is positive. These help me perform equality checks, so that I can implement ‘jumping’ operations.
4. ‘ng’ indicates that the ALU failed to produce an output, and ‘zr’ indicates what error message to print. This is helpful so that an error message can be sent back to the outM.

Question 3: (project3)

If I attempt to assign a value to a register while my load value equals 0, then:

1. The input value will be stored in the register, replacing whatever was stored there previously
2. 🡪 There will be no change to the register
3. This register is clearly full, and it will add the value to the next available register
4. I will erase the value in the register, but I will not store my input

Question 4: (Lecture4)

Why must I end my Hack Machine Language program with an infinite loop?

1. 🡪 Otherwise my stack pointer will continue down past my program, through the rest of the stack
2. My program will not run without it
3. To give the user easier access to the RAM
4. It makes it easier to re-run the program in the future

Question 5: (project5)

Given the following C-Instruction Bus, where will the output for the ALU get stored:

Bus: 1110010110100110

1. The outM (Data Memory)
2. The D register
3. 🡪The A register
4. All the above

Question 6: (project6)

Given the following A-Instruction Bus, find the corresponding notation in Assembly Language: Bus: 0000000100010011

1. @100,010,011
2. M = A+100
3. D = 143
4. 🡪@147

Question 7: (Lecture7)

Which of these is **NOT** a primary benefit for implementing a VM language between the High-Level Language and the Machine Language.

1. You can write one program for many different Hardware platforms.
2. It’s helpful for finding bugs in the code
3. 🡪 it speeds up runtime, and uses less memory
4. compilers in various languages can share the same VM, which allows code sharing and language inter-operability

Question 8: (Lectrue8)

Which of the following statements is True?

1. The Local Segment stores global variables that can be accessed by every function
2. The Argument Segment contains variables that are constant and can never change
3. The Static Segment stores the base addresses of the Keyboard, Stack, and Screen
4. 🡪the That Segment stores pointers, which point to arrays in the heap

Question 9: (Lecture9)

When the Stack Pointer encounters a recursive function, how does it proceed?

1. 🡪The stack pointer will continuously create more and more sub-stacks one after the other within the original stack until either the base case returns a value to argument 0 which cycles back up to the original function caller, or the memory is exceeded, and a stack overflow error is produced.
2. The Stack Pointer will send the function to be handled in the heap, because there is more space for it to be handled there.
3. The Stack Pointer will create each function one on top of the other with the addition of a single register for the return a value. Thus, only the return values are saved with each recursive call, and there is little fear of overloading the memory.
4. The Stack Pointer, uninhibited by the limitations of human beings, is able to skip all the recursive calls, and start at the base case of the recursion, and build itself up. Nothing is saved aside from the altering return value which is stored in argument 0. A stack overflow happens only if the Stack Pointer is stuck in an infinite recursive loop.

Question 10: (Lecture10)

In lesson 10 we learn that each High Level Language has a proper syntax and Grammar format. Which of the following has legal grammar notation for the Jack Language? Note Object is an Identifier, Assume there’s a class called Object with it’s own functions.

1. ‘While’ ‘(‘ ‘*Statement*’ ‘)’ ‘{‘ ‘*Expression’*  ‘;’ ‘}’
2. ‘Let’ ‘*Identifier*’ ‘=’ ‘*Symbol*’ ‘;’
3. ‘Do’ ‘Keyword’ ‘=’ ‘*Term*’ ‘;’
4. 🡪’Object’ ‘.’ ‘*Identifier’* ‘(‘ ‘*Expression*’ ‘)’ ‘;’

Question 11: (Lecture11)

Which of the following Jack Language operations does not necessarily create Labels for the VM code?

1. ‘While’ statements
2. 🡪‘Do’ statements
3. ‘If’ statements
4. ‘For’ statements (this is a trick question as there are no ‘For’ statements in Jack)

Question 12: (project12)

Given an array of size 6. Which of the following ‘Free Blocks’ in the Heap would not be able to store it, and why?

1. A Free Block of size 4, because it doesn’t have enough free blocks to store the 6 values of the array
2. A Free Block of size 6, because although it has space for the 6 values of the array, it needs an extra two segments to store a ‘pointer’ and a ‘size’
3. A Free Block of size 9, because although it definitely has enough space for the Array values, the pointer and the size, by splitting the free block, it’s leaving one segment all by itself that will never be able to initialize a value or defragment it self without wiping the whole RAM. (you have to leave at least two free segments in each free block)
4. 🡪All of the above