**Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

For each of the below questions, write a short sentence or two to express (in your own words) your answer. Keep the answers short, but use complete, correct, English sentences.

If it helps to clarify the questions, feel free to mentally prefix all the questions with the phrase "According to the video…"

1. After you’ve watched all the videos, please answer this question:
Of all the videos that you watched, if you could pick one video to be re-recorded by the instructor outside of class which would you choose? Why?
(Keep in mind the recording outside of class will omit any pauses from the instructor answering student questions, have less hemming and hawing, etc, and generally be more concise)

|  |
| --- |
| < Write your answer here > |

NOTE: There are no Viewing Quiz questions for the following videos. Instead, you should fill out the following video outlines.
NOTE #2: There are [directions on the course website that explain how to outline the videos](http://faculty.cascadia.edu/mpanitz/Courses/BIT142/Homeworks/Outlining/index.html). These direction videos are pretty short, so please do watch them!

**Outline for “How To Calculate Your Grade”**

**File: 07\_How\_To\_Calculate\_Your\_Grade.mp4**

* First major topic (replace this with your outline)

**Outline for “Suggested Workflow”**

**File: 08\_Workflow.mp4**

* First major topic (replace this with your outline)

**VIDEO: Stacks: ADT, Examples**

1. What are some example of simple/primitive data types? Where is the memory needed to store these items allocated?

|  |
| --- |
|  |

1. What does ADT stand for? What is the purpose of ADTs in programming?

|  |
| --- |
|  |

1. Fill in the blank: “A stack is what’s called a \_\_(1)\_\_ data structure. Meaning that the \_\_\_(2)\_\_\_\_ thing that you’ve added to the stack is the first thing you’re going to get back out”

|  |
| --- |
|  |

1. In a “pure stack” which item(s) are you allowed to examine/get?

|  |
| --- |
|  |

1. What does the “push” method do?

|  |
| --- |
|  |

1. What does the “pop” method do?

|  |
| --- |
|  |

1. *Briefly,* intuitively summarize what a call stack is, and why it must be a stack (why does it make sense that the program will only access the top-most stack frame)?
(The purpose of this question is to make sure that you’ve seen a couple of examples of how to use a stack, and NOT to be able to explain this particular example in detail!)

|  |
| --- |
|  |

1. *Briefly,* intuitively summarize how does the ‘Undo’/’Redo’ feature use a stack(s)?
(The purpose of this question is to make sure that you’ve seen a couple of examples of how to use a stack, and NOT to be able to explain this particular example in detail!)

|  |
| --- |
|  |

1. How is “Reverse Polish Notation” set up? Give a quick example:

|  |
| --- |
|  |

1. How can a stack be used to parse/calculate a mathematical expression written using “Reverse Polish Notation”?

|  |
| --- |
|  |

1. Why is “Reverse Polish Notation” sometimes called a “postfix” notation? What sort of notation we normally used? What is a “prefix” notation?

|  |
| --- |
|  |

**VIDEO: Stacks: API, Implementation**

1. What does the acronym API stand for? In a nutshell, what is an API?

|  |
| --- |
|  |

1. What we will use as the API for the **push** function?
(Include the return value, function/method name, any parameters, and describe any possible errors that might happen.)

|  |
| --- |
|  |

1. What we will use as the API for the **pop** function?
(Include the return value, function/method name, any parameters, and describe any possible errors that might happen.)

|  |
| --- |
|  |

1. What does the **peek** function do (and how does it differ from pop)? What we will use as the API for the **peek** function?
(Include the return value, function/method name, any parameters, and describe any possible errors that might happen.)

|  |
| --- |
|  |

1. What we will use as the API for the **isEmpty** function?
(Include the return value, function/method name, any parameters, and describe any possible errors that might happen.)

|  |
| --- |
|  |

1. What C# data type(s) will we use to implement our stack class?

|  |
| --- |
|  |

1. For our implementation, what does the ‘top’ instance variable mean?
(If it has the value -1 what does that mean? If ‘top’ has the value 0 what does that mean?)

|  |
| --- |
|  |

1. Intuitively, describe how our push method will work. Make sure that you describe all of the following:
How does the method detect that the stack is full?
How does the method detect that the stack is empty?
If the stack is not full, what is done?

|  |
| --- |
|  |

**VIDEO: Queue: API, Implementation**

1. A queue is referred to as a \_\_\_\_\_\_ data structure (hint: the answer is 1 word, 4 letters long, and it’s an acronym)

|  |
| --- |
|  |

1. When we add a new item to the queue, which end is it placed at?
When we remove an new item to the queue, which end is it removed from?

|  |
| --- |
|  |

1. What is another word that means “add” to a queue?
What is another word that means “remove” from a queue?

|  |
| --- |
|  |

1. What is the problem with a simple, naïve implementation of a queue that uses an array (i.e., what is the problem with a non-circular implementation)?

|  |
| --- |
|  |

1. If you did use a simple, naïve implementation of a queue (as an array), why is it bad to make space within the queue by shifting down all the elements in the array?
What is the running time to shift all the elements down by one?

|  |
| --- |
|  |

1. What is the basic idea behind a **circular queue**? What happens when the reference to the back of the queue reaches the end of the array (assuming that there’s empty space at the front of the array)?

|  |
| --- |
|  |

1. C#’s only ternary operator (operator that takes 3 operands) is the ?: operator. Explain (concisely) what the following line of code does (the parentheses have been color-coded so they’re easier to visually match.
back = **(** **(**back + 1**)** == items.Length **)** ? 0 : **(**back + 1**)**;

|  |
| --- |
|  |

1. Re-write the following code snippet to use an equivalent if…else statement:
back = **(** **(**back + 1**)** == items.Length **)** ? 0 : **(**back + 1**)**;

|  |
| --- |
|  |

1. Deqeue is very similar to how Enqueue is written. Based on what’s in the slide and what was described in the audio, write out the Dequeue function here.
(Note – you are not required to get this to compile & run 100% correctly – please feel free to “code this up” here in Word. The main objective is to make sure that you followed the description overall)

|  |
| --- |
|  |