Alex Moore

Live Coding Activity - Teaching

Mini-CreatePracticeTask

Learning from Four Student Exemplars: Procedures (Rows 4, 5, and 6)

Note: The program excerpts below come from students' practice Mini-Create Tasks. They are provided as a means of supporting students with additional models as they revise their Mini-Create Tasks.

Be sure to check out the companion to this handout, Learning from Four Student Exemplars: Lists (Rows 2 and 3) in Google Classroom->Supports!
https://classroom.google.com/c/MTY0Njk2NTA2ODg4/m/Mjk3MTk0NDg5ODA2/details

Anabel

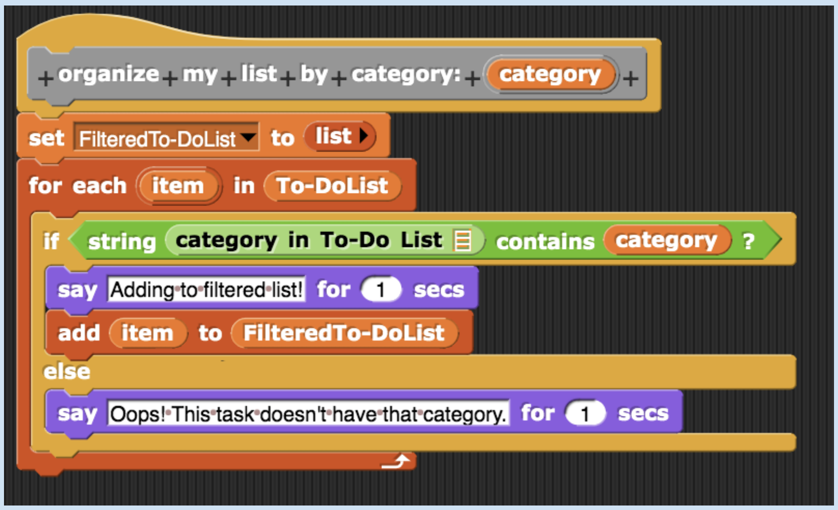
3c. Capture and paste two program code segments you developed during the administration of this task that contain a student-developed procedure that implements an algorithm used in your program and a call to that procedure.
Approx. 200 words (for all subparts of 3c combined, exclusive of program code)

i. The first program code segment must be a student-developed procedure that:

☐ Defines the procedure's name and return type (if necessary)

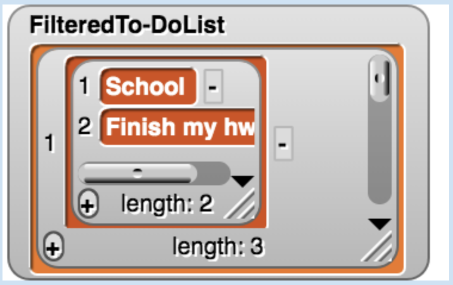
☐ Contains and uses one or more parameters that have an effect on the functionality of the procedure

☐ Implements an algorithm that includes sequencing, selection, and iteration

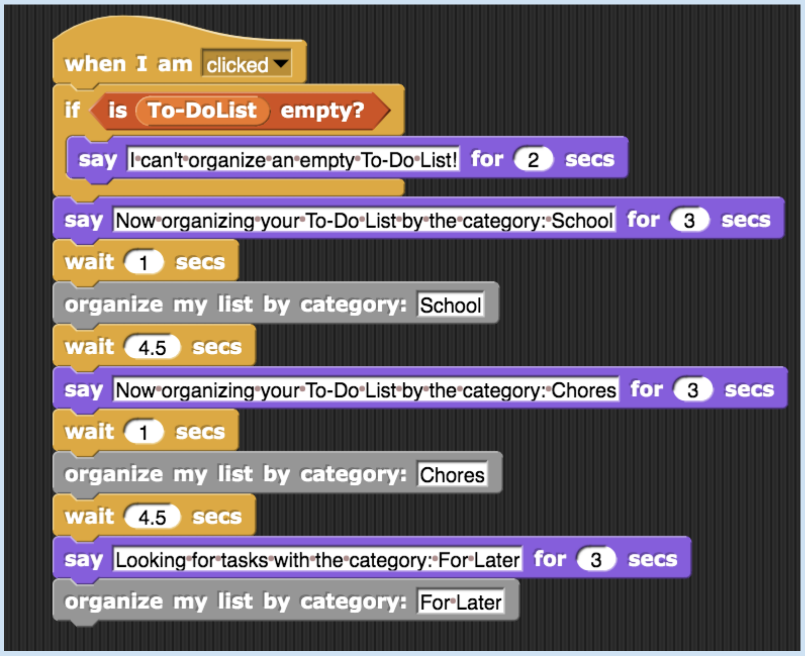


```
+organize+my+list+by+category:+(category)+
set FilteredTo-DoList to (list)
for each (item) in To-DoList
    if < string (category in To-Do List) contains (category) ? >
        say [Adding to filtered list!] for (1) secs
        add (item) to FilteredTo-DoList
    else
        say [Oops! This task doesn't have that category.] for (1) secs
```

Live Coding Notes - Avoid expert bias. Make is accessible and inviting. Include tougher questions as well. Questions to be asked while coding follow: Is category an argument or a parameter? Where does category appear elsewhere in this procedure? If category had a value of "age," what would be the value of category on line 4? Is FilteredTo-DoList local or global? When does a for each loop end? What is a string? Where is this string coming from? What does contains do? Let's test this out with a real list. In what case does it evaluate to TRUE or FALSE? If the Boolean expression is TRUE, do we execute lines 5 and 6 or just 5? Why are we adding item to FilteredToDoList? When does the else code run? Can FilteredTo-DoList variable be accessed outside of the procedure? Explain sentence the parameters, functionality, and return value - if any - of this procedure. What doesn't make sense to you?



ii. The second program code segment must show where your student-developed procedure is being called in your program.



```
when I am clicked
if is To-DoList empty?
    say I can't organize an empty To-Do List! for 2 secs
say Now organizing your To-Do List by the category: School for 3 secs
wait 1 secs
organize my list by category: School
wait 4.5 secs
say Now organizing your To-Do List by the category: Chores for 3 secs
wait 1 secs
organize my list by category: Chores
wait 4.5 secs
say Looking for tasks with the category: For Later for 3 secs
organize my list by category: For Later
```

This is the script under the "Filter My List" button.

Then, provide a written response that does both of the following:

iii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program

The procedure filters the user's To-Do List by 3 different categories ("School", "Chores", and "For Later"). If the To-Do List does contain the category, the user will receive a message saying it was added to the new filtered list titled "FIlteredTo-DoList" while if it's not, they will be told that the task does not contain the category. This makes the program more efficient for the user so they do not have to look through their whole "To-DoList" searching for tasks under the same category.

iv. Explains in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

The “organize my list by category:___” block works by taking a user input called category, that is passed through every item of the “To-DoList”. When the code is run, the user will receive a message “Now organizing your To-Do List by the category:___” based on whatever text was written in the block’s input slot. For each item in the “To-DoList”, if the item includes the category, the message “Adding to filtered list!” will appear on the stage and that item will be added to a new list which is shown on the stage titled “FilteredTo-DoList”. If the item does not contain the category, the message “Oops! This task doesn’t have that category.” will appear and the item will not be added to “FilteredTo-DoList”.

3d. Provide a written response that does all three of the following:
Approx. 200 words (for all subparts of 3d combined)

i. Describes two calls to the procedure identified in written response 3c. Each call must pass a different argument(s) that causes a different segment of code in the algorithm to execute.

First call:

The first call organizes the list by the category “School”. The argument in this call is the category name, “School”.

Second call:

Another call takes in the argument “For Later”. The argument in this call is the category name “For Later”.

ii. Describes what condition(s) is being tested by each call to the procedure

Condition(s) tested by the first call:

Whether the word “School” appears in the category of an item in the general list titled “To-DoList” (through the use of an if-else statement).

Condition(s) tested by the second call:

Whether the phrase “For Later” appears in the category of an item in the general list titled “To-DoList” (through the use of an if-else statement).

iii. Identifies the result of each call

Result of the first call:

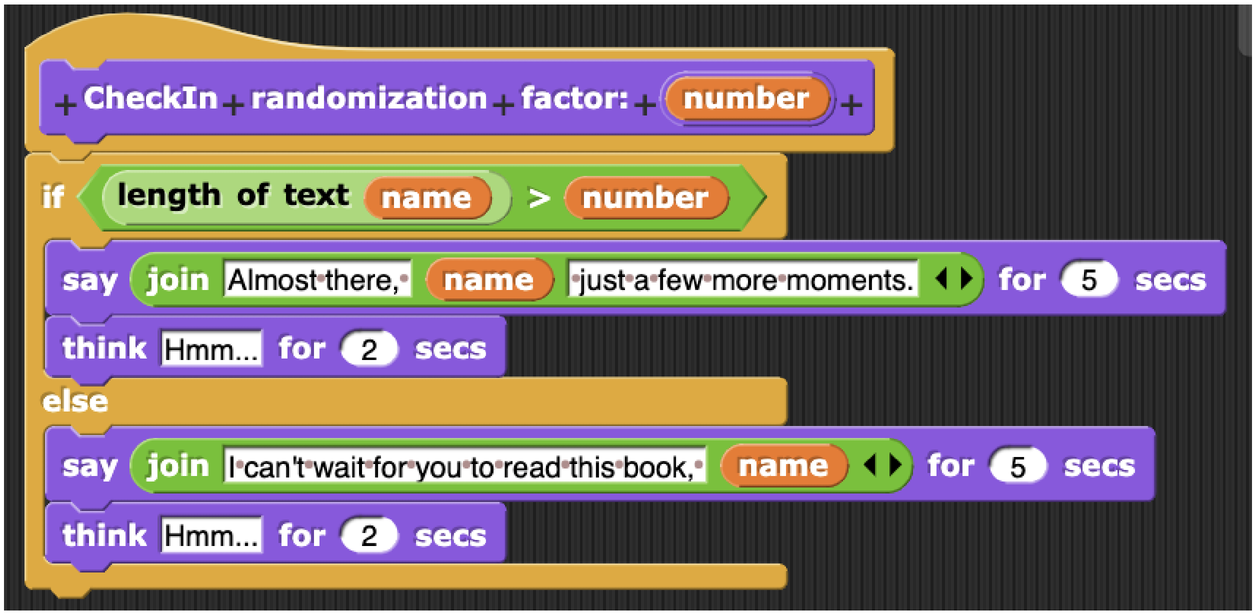
According to the items already entered in the "To-DoList", 3 items are added to "FilteredTo-DoList" which shows the items that passed the check for having the category name "School."

Result of the second call:

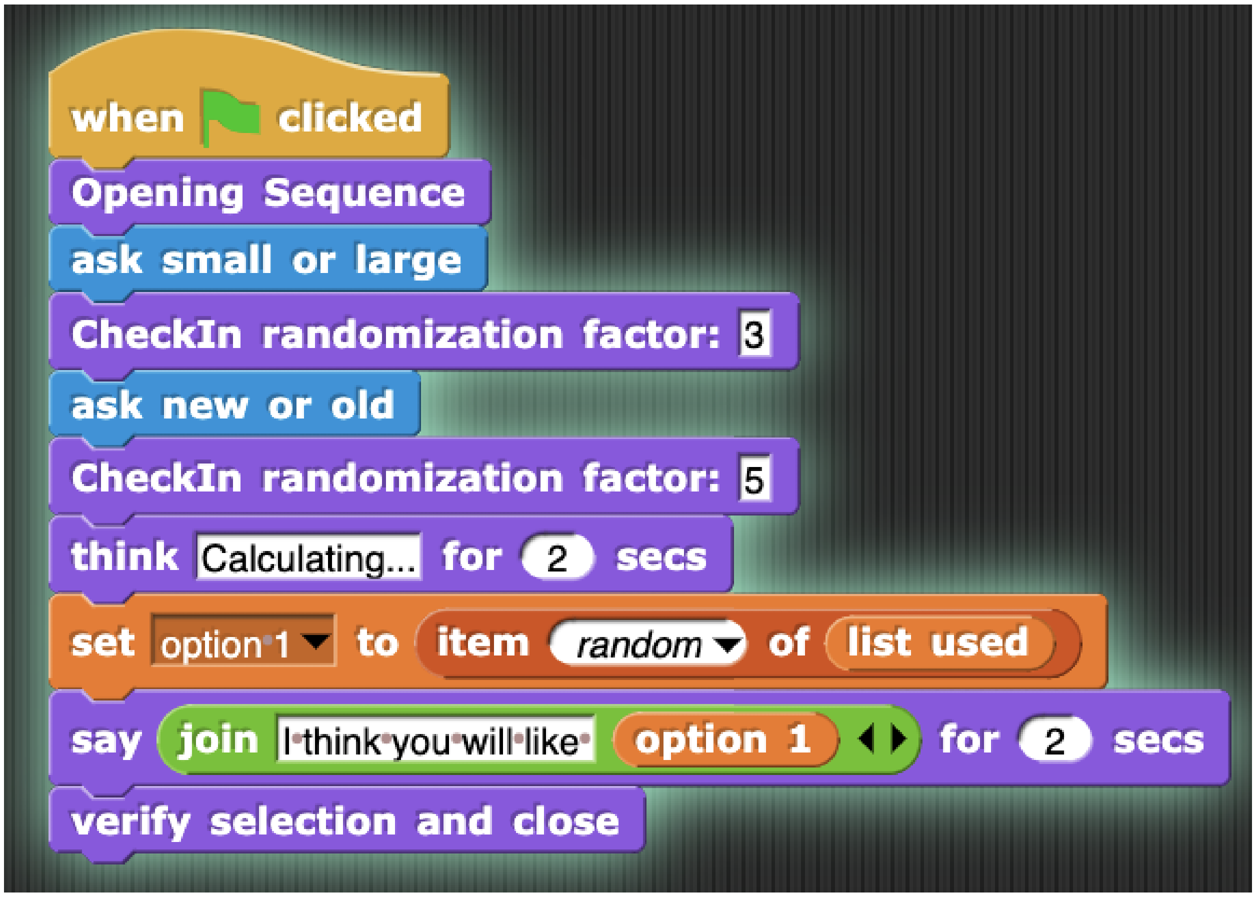
According to the items already entered in the "To-DoList", 0 items are shown in "FilteredTo-DoList" because none of the items contained the category name "For Later".

Ashley

3c.



```
+ CheckIn + randomization + factor: + (number) +
if < (length of text (name)) > (number) >
    say (join [Almost there, ] (name) [ just a few more moments.] ◀▶) for (5) secs
    think [Hmm...] for (2) secs
else
    say (join [I can't wait for you to read this book, ] (name) ◀▶) for (5) secs
    think [Hmm...] for (2) secs
```



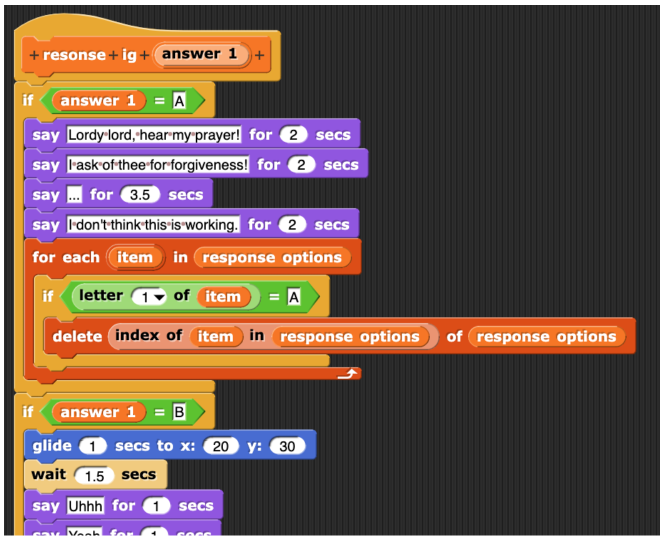
```
when [flag] clicked
Opening Sequence
ask small or large
CheckIn randomization factor: 3
ask new or old
CheckIn randomization factor: 5
think Calculating... for 2 secs
set option 1 to item random of list used
say join I think you will like option 1 for 2 secs
verify selection and close
```

The identified procedure, based on how many letters are in the user's name, provides a generated greeting indicating that the book recommendation is on its way. In this way, it is more personalized and friendly. This algorithm works by the programmer implementing a "randomization factor" block variable. If this number is greater than the number of letters in the user's name, the BookWorm will say something different than otherwise. The factor is different the two times it is in the program so that the block may behave differently.

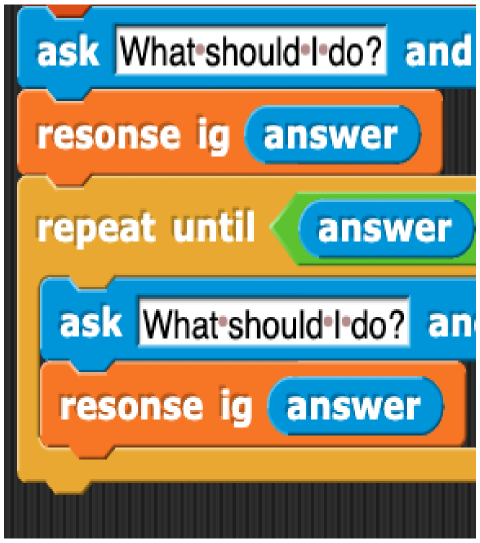
3d. If the randomization factor is 3 and the number of letters in the user's name is 5, the sprite will say "Almost there, [name], just a few moments more" and then think "hmm…" If the randomization factor is 5 and the number of letters in the user's name is 5, the sprite will say "I can't wait for you to read this book" and then think "hmm…" In both situations, both the randomization factor and the number of letters in the user's name are being tested. Depending on these variables, the sprite will produce a different conversation.

Esme

3c. Capture and paste two program code segments you developed during the administration of this task that contain a student-developed procedure that implements an algorithm used in your program and a call to that procedure.



```
+ resonse + ig + (answer 1) +
if <(answer 1) = A>
    say [Lordy lord, hear my prayer!] for (2) secs
    say [I ask of thee for forgiveness!] for (2) secs
    say [...] for (3.5) secs
    say [I don't think this is working.] for (2) secs
    for each (item) in (response options)
        if <(letter (1) of (item)) = A>
            delete (index of (item) in (response options)) of (response options)
if <(answer 1) = B>
    glide (1) secs to x: (20) y: (30)
    wait (1.5) secs
    say [Uhhh] for (1) secs
```



```
ask [What should I do?] and
resonse ig (answer)
repeat until <(answer)
    ask [What should I do?] an
    resonse ig (answer)
```

Approx. 200 words (for all subparts of 3c combined, exclusive of program code)

i. The first program code segment must be a student-developed procedure that:

☐ Defines the procedure's name and return type (if necessary)

☐ Contains and uses one or more parameters that have an effect on the functionality of the procedure

☐ Implements an algorithm that includes sequencing, selection, and iteration

The first example is one of the "response options" in-depth. If you were to select option A, the second example breaks down what the sprite would say until you reach answer C, the final answer. In other words, it's taking "Answer 1" as the input and is telling the sprite to output different text (depending on what the input is). Once it does that, the code will iterate over the list to find and delete that 'response option' from the list, as it is no longer a viable option to try.

ii. The second program code segment must show where your student-developed procedure is being called in your program.

The second image asks you for an input, and then once that is provided, it calls the response function illustrated in the first image using the input. If the input is C, the program ends but if it's anything else, the code will restart and ask you to try again until C is inputed.

Then, provide a written response that does both of the following:

iii. Describe in general what the identified procedure does and how it contributes to the overall functionality of the program

The identified procedure on the right is asking you [the cod-ee] for an input that will result in the sprite either moving or speaking through the response function. This contributes to the overall functionality as it keeps the program running until "C" is eventually inputted. The identified procedure on the left is one of the 'response options' opened up and dissected. It shows what happens if you choose the incorrect option; you will interact with the sprite for a moment then you will be asked to try again. By asking you to try again, you are learning what *not* to do again, while also getting closer to the end goal of the algorithm. It's contributing to the functionality of the code as it's giving a visual representation of the possible inputs for the program itself.

iv. Explain in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

Essentially, the image on the left will use answer 1 as an input, and depending on what 'answer 1' is, it will provide different outputs, resulting in the sprite taking up a tone or a new location! Since in this case, 'answer 1' was A and not C, it will then delete the 'answer 1' (A) in the list of available 'response options' above, as the cod-ee used it and it didn't work.

3d. Provide a written response that does all three of the following:
Approx. 200 words (for all subparts of 3d combined)

i. Describes two calls to the procedure identified in written response 3c. Each call must pass a different argument(s) that causes a different segment of code in the algorithm to execute.

First call:

Once the program asks the cod-ee "what should I do?" for the first time, the procedure is called using the cod-ee's input. What is happening is that you [the cod-ee] are being asked a series of questions by the sprite, in which you will answer, prompting the sprite [bus] to either speak or move.

Second call:

The second call of the function is inside of a "repeat until" block. What this is doing is prompting the algorithm to run until the cod-ee inputs the letter "C". Until this happens, the sprite will speak and move, however the program will not end, as C is the only viable solution. Because the 'response options' are deleted from the list once used, the cod-ee is forced to provide a different input.

ii. Describes what condition(s) is being tested by each call to the procedure

Condition(s) tested by the first call:

"Answer 1" is the parameter that determines which segment of code is run based on which input is provided. If statements check if "answer 1" is A, B, or C, and depending on what the if statement finds, it will provide different outputs under different conditions.

Condition(s) tested by the second call:

“Answer 1” in the second call, similar to the first one, will check if the input is A, B, or C. The second call differs from the first because when an input is used, it gets deleted from the list of options.

iii. Identifies the result of each call

Result of the first call:

The first call results will vary depending on what the input is. If the input is A or B, the sprite [bus] will move and/or speak, however the program will keep running. If the input is C, the program will end.

Result of the second call:

If the first call’s input was anything other than C, the second call will come into effect. You will be asked to try again, and this time, the output will vary as the input changes from what it was initially supposed to be (which, in this case, was C). In this scenario, the output must vary from inputting C as the ‘response options’ list will no longer have the option chosen initially.

Pan

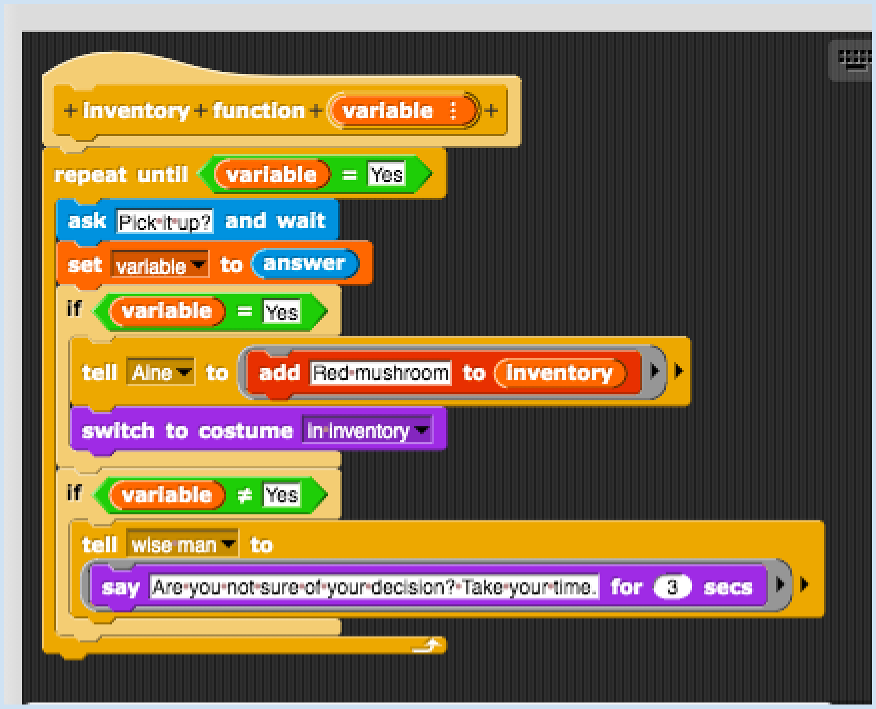
3c. Capture and paste two program code segments you developed during the administration of this task that contain a student-developed procedure that implements an algorithm used in your program and a call to that procedure.
Approx. 200 words (for all subparts of 3c combined, exclusive of program code)

i. The first program code segment must be a student-developed procedure that:

☐ Defines the procedure’s name and return type (if necessary)

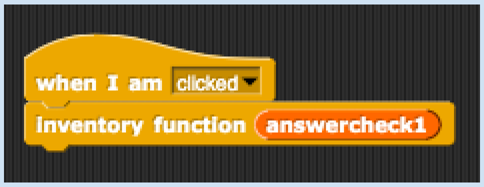
☐ Contains and uses one or more parameters that have an effect on the functionality of the procedure

☐ Implements an algorithm that includes sequencing, selection, and iteration



```
+ inventory + function + (variable :) +
repeat until <(variable) = Yes>
  ask Pick·it·up? and wait
  set variable to (answer)
  if <(variable) = Yes>
    tell Aine to (add Red·mushroom to (inventory)) ▸
    switch to costume In·inventory
  if <(variable) ≠ Yes>
    tell wise man to
      (say Are·you·not·sure·of·your·decision?·Take·your·time. for 3 secs) ▸
```

ii. The second program code segment must show where your student-developed procedure is being called in your program.



```
when I am clicked
inventory function (answercheck1)
```

Then, provide a written response that does both of the following:

iii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program

The identified procedure allows the player to add an item to their inventory simply by pressing it, and assuring the program that they do want to add it. This feature is essential to progressing the storyline of the RPG, as the game cannot end without adding items to your inventory.

iv. Explains in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

The function includes an input labeled “variable”. Each item that can be picked up has a different input that matches the one they’ve been set to at the beginning of the program using the [set __ to []] block. This input acts as the answer that the player will input when asked if they want to pick this item up. This is to ensure that the program runs smoothly in terms of the “repeat” block in the function working effectively. The function will ask “Pick it up?”, and if the player inputs “Yes”, then the item will switch to a costume that makes it seem like it’s not on the stage anymore and “Red mushroom” will be added to the inventory list. If the player does not input “Yes”, the program will say to try again until “Yes” is inputted.

3d. Provide a written response that does all three of the following:
Approx. 200 words (for all subparts of 3d combined)

i. Describes two calls to the procedure identified in written response 3c. Each call must pass a different argument(s) that causes a different segment of code in the algorithm to execute.

First call:

If the player inputs “Yes”, the item that was clicked will switch to a costume that will make it seem as if it’s not on the stage anymore and it’s gone into your inventory, and the list item “Red mushroom” will go into inventory.

Second call:

If the player inputs “No”, an NPC will say to try again, and the function will repeat itself until the player inputs “Yes”, as a way to encourage the player to play the game correctly.

ii. Describes what condition(s) is being tested by each call to the procedure

Condition(s) tested by the first call:

If “Yes” is inputted.

Condition(s) tested by the second call:

If “No” is inputted.

iii. Identifies the result of each call

Result of the first call:

The item that was clicked will switch to a costume that will make it seem as if it’s not on the stage anymore and it’s gone into your inventory, and the list item “Red mushroom” will go into inventory.

Result of the second call: