1 Create Performance Task

"It always takes longer than you expect, even when you take into account Hofstadter's Law."

— Hofstadter's Law

Chapter Goals

- General requirements
- Program requirements
- Programming language requirements
- Preprogramming checklist
- How to get input from the user
- Iteration
- How to fill up a list
- Level of complexity of a program
- Procedures with parameters
- Written question 3a–d
- Sample Create Performance Task One
- Sample Create Performance Task Two
- Sample Create Performance Task Three
- Sample Create Performance Task Four
- Sample Create Performance Task Five
- Sample Create Performance Task Six

General Requirements

The Create Performance Task is worth 30 percent of your AP exam. If you pay careful attention to the prompts and rubric, no points should be lost on this section. The prompts and rubric can change slightly, and the current version can be found on *apcentral.collegeboard.org*. Make sure you are using the most current version of the rubric.

The table below shows the percentage of students on the 2022 AP CSP exam who earned ANY points on the following Create project sections.

Create Section	Percentage of Students Earning ANY Points
Program Purpose and Function	51%
Data Abstraction	70%
Managing Complexity	36%
Procedural Abstraction	51%
Algorithm Implementation	47%
Testing	40%

Programming is a collaborative and creative process that brings ideas to life through the development of software. Programs can help solve problems, enable innovations, or express personal interests. In this performance task, you will be developing a program of your choice. Your development process should include iteratively designing, implementing, and testing your program. You are strongly encouraged to work with another student in your class for parts of this program.

WARNING: You **can** collaborate with your partner(s) on the development of the program only. The written response and the video that you submit for this performance task must be completed **individually**, without any collaboration with your partner(s) or anyone else.

Your teacher must guarantee at least **12 hours** of class time and unlimited out of class time to complete and submit the following:

- Final program code (created independently or collaboratively)
- A video that displays the running of your program and demonstrates functionality you developed (created independently)
- Written responses (a–d) to all the prompts in the performance task (created independently)

Your video must show:

- Input to your program
- At least one aspect of the functionality of your program
- Output produced by your program

Your video must NOT show:

- Any distinguishing information about yourself
- Voice narration (although text captions are encouraged)

Your video must meet the following specifications:

- be stored in .mp4, .wmv, .avi, or .mov format
- be no more than 1 minute in length
- be no more than 30 MB in file size

Program Requirements

Your program must demonstrate a variety of capabilities and implement several different language features that, when combined, produce a result that cannot be easily accomplished without computing tools and techniques. Your program should draw upon a combination of mathematical and logical concepts, such as use of numbers, variables, mathematical expressions with arithmetic operators, logical and Boolean operators and expressions, decision statements, iteration, and collections.

Your program must demonstrate:

- Instructions for **input** from one of the following:
  - The user (including user actions that trigger events)
  - A device
  - An online data streams
  - A file
- Use of at least one **list** (or other collection type) to represent a collection of data that is stored and used to manage program complexity and help fulfill the programs purpose. A list can be an Array, ArrayList, LinkedList, etc.
- At least one **procedure** (abstraction, method, subroutine) that contributes to the program's intended purpose, where you have defined:
  - The procedure's name
  - The return type (if necessary)
  - **One or more parameters**

- An algorithm that includes **sequencing, selection, and iteration**, that is in the body of the selected procedure
- Calls to your student-developed procedure
- Instructions for output (tactile, audible, visual, or textual) based on input and program functionality

There is NO designated programming language for AP Computer Science Principles. Students may choose a programming language learned while taking the course to complete the task, or they may select a different programming language that they are familiar with outside of class.

Sample languages that can be used for the Create Performance Task:

JAVA	Python	SNAP!	Scratch
MobileCSP	ALICE	C++	C

WARNING: HTML is NOT an accepted language for the Create Performance Task.

As long as the language is able to create a list and call a procedure that takes in a parameter, you can use it. There is no scoring difference between using a low-level language or a high-level language.

Preprogramming Checklist

When selecting the program focus, students should ensure that their program will be sophisticated enough to integrate mathematical and logical concepts, develop abstraction, use a list, and implement algorithms.

A 500-line program will not necessarily score higher than a 30-line program. In fact, a 500-line program that does not iterate or have a procedure (can be called abstractions, functions, or methods) that takes in a parameter will score lower than the 30-line program if the 30-line program contains all the correct elements.

The programming topic is completely up to you and should be something that interests you. However, you need to account for the limited number of class time hours. If you want to write the new version of *Halo,* go for it. However, *Halo* will not be able to be written in the provided 12 hours. Think of the scope of your program before diving in.

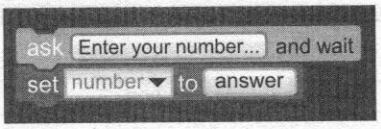
While writing the program, you should be aware that your end program requires instructions for input, a list, a procedure that takes in a parameter, and an algorithm that includes sequencing, selection, and iteration.

Programming Checklist to Hit the Rubric

Does your program have at least one list (a list can also be an array, database, or set) that is integral to the function of your program?	yes/no
Are the data stored in the list vital to the function of the program?	yes/no
Is the program showing how the data are loaded into the list?	yes/no
Are the data in the list accessed by iteration (not directly accessed by index)?	yes/no
Does your program take in and use input from the user, device, or online data stream or a file?	yes/no
Does your program contain a procedure that takes in parameters? Are those parameters being used?	yes/no

Some Examples of How to Get Input from the User (JAVA and SNAP!)

Sample Text-Based Program (JAVA)
The import statement needs to be the first line in your program and documented in your program.
	`import java.util.*;`
	`Scanner scan = new Scanner(System.in);`
	`System.out.println("Enter a number");`
	`int num = scan.nextInt();`
	Look up the scanner API to read in different types of data. Put in comments that you used the Scanner Library.

Sample Graphical Language	

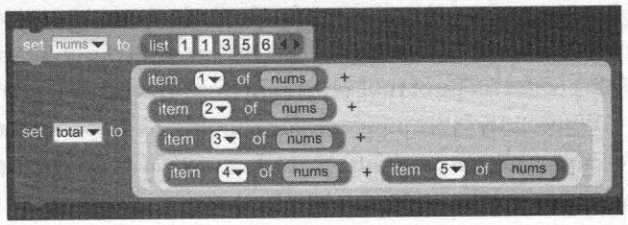
For information on getting input for any computer language I recommend the website *stackoverflow.com.*

Example:
To determine how to get input using the language Python, I searched the following "input from user in python stack overflow."

Examples and Nonexamples of Iteration

Iteration	NOT Iteration

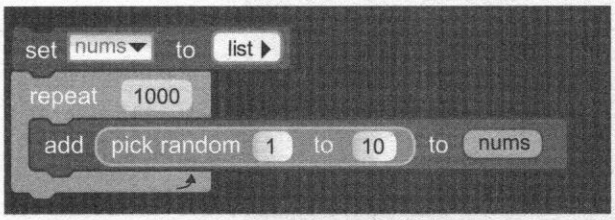
`int [] nums = { 1, 1, 3, 5, 6}`
`int total = 0;`
`for (int item : nums)`
`{`
`  total = total + item;`
`}`
	`int [] nums = { 1, 1, 3, 5, 6}`
	`total = nums[0] + nums[1] + nums[2] + nums[3] + nums[4];`

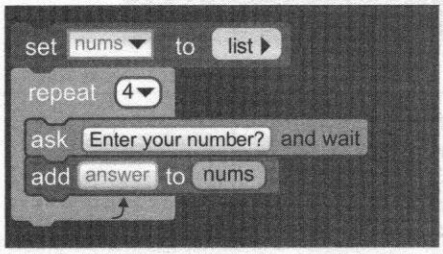
Various Ways to Fill Up a List

```
int[] nums = {1,1,3,5,6};
```

```
int [] nums = new int [5];
nums[0] = 1;
nums[1] = 1;
nums[2] = 3;
nums[3] = 5;
nums[4] = 6;
```




```
int[] nums = new int[1000];
for(int x = 0 ; x < nums.length ; x++)
{
 nums[x]= (int)(Math.random() * 100) + 1;
}
```



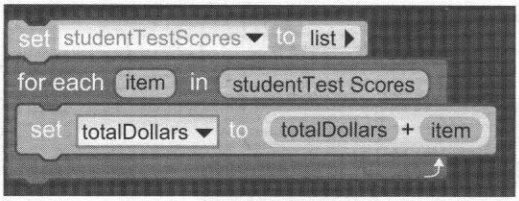

```
int[]nums = new int(10];
Scanner scan = new Scanner(System.in);
for(int x = 0 ; x < nums.length ; x++)
{
  System.out.println("Enter a number");
  nums[x] = scan.nextInt();
}
```

Reducing the Level of Complexity of Your Program

Programs are complicated. Every step, however small it may be, should be taken to reduce the level of complexity of your program.

How does a data structure reduce the level of complexity of a program?

The name of the list should represent what is being held in the list. For example, if your list is holding student test scores, a few good names for the list would be testScore or studentTestScores. By using these names, little doubt is left as to what they hold.	`int [] testScores = new int[8]`

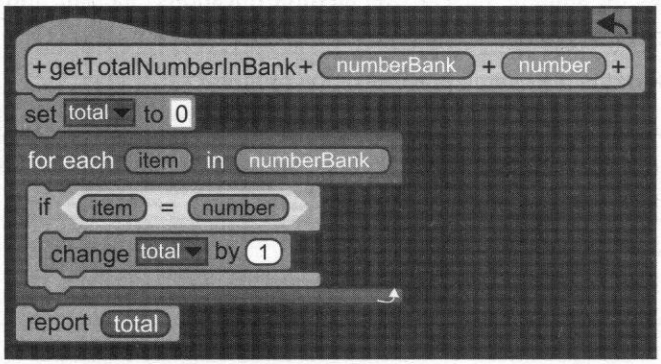
A list further reduces the complexity of your program by putting data in an accessible way. For example, if I wanted to add up all the elements of a list, I can access them directly with a loop and index.	`// Assume the array has 1000 elements`
	`int [] dollarsMadeForAllStores`
	`for ( int item: dollarsMadeForAllStores)`
	`{`
	`totalDollars = totalDollars + item;`
	`}`
	 

Without a list, each test score would need its own name. To the right is an example with only 8 tests. The longer the code is, the more likely it is an error will appear. For example, in the code to the right, I put a space between test and 8. This would result in a syntax error and would be avoided had I used a list.	`int test1 = 80;`
	`int test2 = 90;`
	`int test3 = 100;`
	`int test4 = 20;`
	`int test5 = 80;`
	`int test6 = 90`
	`int test7 = 70`
	`int test8 = 65`
	`int total = test1 + test2 + test3 + test4 + test5 + test6 + test7 + test8;`

WARNING: Do not try to force a list where it does not belong. Your list must have more then one element in it. The rubric states the list must reduce the level of complexity of code. If the list contains one element, it may increase the level of complexity of your code.

Procedure with Parameters

Your program must contain at least one procedure that contains at least one parameter that has an effect on the functionality of the procedure.

I want a procedure to check a list for a number entered by the user. The program will not know what number to look for until run time.	The getTotalNumberInBank has two parameters. numberBank is a list that will be iterated through, and number is the number that is being searched for in the list. Notice this procedure has both iteration and selection. 

	The range procedure will return if the user-entered range is less than the actual range calculated in the procedure. The procedure has two parameters. The first parameter is the integer array, and the second parameter is the range.
	`public static boolean range( int[] numbers, int range)`
	`{`
	`int max = numbers[0];`
	`int min = numbers[0];`
	`for(int x = 0 ; x < numbers.length; x++)`
	`{`
	`if(max > numbers[x])`
	`max = numbers[x];`
	`if(min < numbers[x])`
	`min = numbers[x];`
	`}`
	`if( range < max – min)`
	`return true;`
	`else`
	`return false;`
	`}`

WARNING: Do NOT put parameters into your procedure that do not play a part in your program.

Written Question 3a–d

Answer all the boxes. Do not focus too much on your grammar—the important part is actually answering the question. Never give up and leave a box blank. The word count for all prompts combined is 750 words but can go as high as 850 words. Program code does not count as words. **Collaboration is NOT allowed on the written responses.**

3a. You can write approximately 150 words for all subparts of 3a combined. To get the five points, the student must answer all three questions (3ai, 3aii, and 3aiii) correctly. No partial credit is given.

i. Describe the overall purpose of the program.

ii. Describe what functionality of the program is demonstrated in the video.

iii. Describe the input and output of the program demonstrated in the video.

3b. Capture and paste two program segments you developed during the administration of this task that contain a list (or other collection type, including arrays) being used to manage complexity in your program.

You can write approximately 200 words for all subparts of 3b combined, exclusive of program code. To get the five points, the student must answer all four questions (3bi, 3bii, 3biii, and 3biv) correctly. No partial credit is given.

i. The first program code segment must show how data have been stored in the list.

ii. The second program code segment must show the data in the same list being used, such as creating new data from existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

Then provide a written response that does all three of the following:

iii. Identifies the name of the variable representing the list being used in this response.

iv. Describes what the data contained in the list are representing in the program.

v. Explains how the named, selected list manages complexity in the program code by explaining why the program code could not be written, or how it would be written differently, without using the list.

3c. Include 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. You can use approximately 200 words for all subparts of 3c combined, not including program code.

WARNING: Built-in or existing procedure and language structures, such as event handlers and main methods, are NOT considered student developed. To get the five points, the student must answer all three questions (3ci, 3cii, and 3ciii) correctly. No partial credit is given.

i. The first program 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 influence the functionality of the procedure
- Implements an algorithm that includes sequencing, selection, and iteration

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

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.

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.

3d. Provide a written response that does all three of the following (using approximately 200 words for all subparts of 3d combined). To get the five points, the student must answer all three questions (3di, 3dii, and 3diii) correctly. No partial credit is given.

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:

Second call:

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

Condition(s) tested by the first call:

Condition(s) tested by the second call:

iii. Identifies the result of each call

Result of first call:

Result of second call:

Create Performance Task Rubric from College Board

Each box is worth five points. There is no partial credit.

Reporting Category	Scoring Criteria	Decision Rules
Row 1 **Program Purpose and Function** **(0–1 points)** **4.A**	The video demonstrates the running of the program including: • *input* • *program functionality* • *output* **AND** The written response: • describes the overall *purpose* of the program. • describes what functionality of the program is demonstrated in the video. • describes the input and output of the program demonstrated in the video.	**Consider ONLY the video and written response 3a when scoring this point.** **Do NOT award a point if the following is true:** • The video does not show a demonstration of the program running (screenshots or storyboards are not acceptable and would not be credited).

Continued

Row 2 **Data Abstraction** **(0–1 points)** **3.B**	The written response: • includes two *program code segments*: - one that shows how *data has been stored in this list* (or other *collection type*). - one that shows the data in this same *list being used* as part of fulfilling the program's purpose. • identifies the name of the variable representing the list being used in this response. • describes what the data contained in this list is representing in the program.	**Consider ONLY written response 3b when scoring this point.** **Requirements for program code segments:** • The written response must include two clearly distinguishable program code segments, but these segments may be disjointed code segments or two parts of a contiguous code segment. • If the written response includes more than two code segments, use the first two code segments to determine whether or not the point is earned. **Do NOT award a point if the following is true:** • The use of the list is trivial and does not assist in fulfilling the program's purpose.
Row 3 **Managing Complexity** **(0–1 points)** **3.C**	The written response: • includes a program code segment that shows a list being used to manage complexity in the program. • explains how the named, selected list manages complexity in the program code by explaining why the program code could not be written, or how it would be written differently, without using this list.	**Consider ONLY written response 3b when scoring this point.** **Responses that do not earn the point in row 2 may still earn the point in this row.** **Do NOT award a point if any one or more of the following is true:** • The code segments containing the lists are not separately included in the written response section (not included at all, or the entire program is selected without explicitly identifying the code segments containing the list). • The written response does not name the selected list (or other collection type). • The use of the list is irrelevant or not used in the program. • The explanation does not apply to the selected list. • The explanation of how the list manages complexity is implausible, inaccurate, or inconsistent with the program. • The solution without the list is implausible, inaccurate, or inconsistent with the program. • The use of the list does not result in a program that is easier to develop, meaning alternatives presented are equally complex or potentially easier. • The use of the list does not result in a program that is easier to maintain, meaning that future changes to the size of the list would cause significant modifications to the code.

Continued

Row 4 **Procedural Abstraction** **(0–1 points)** **3.B**	The written response: • includes two program code segments: - one showing a *student-developed procedure* with at least one *parameter* that has an effect on the functionality of the procedure. - one showing where the student-developed procedure is being called. • describes what the identified procedure does and how it contributes to the overall functionality of the program.	**Consider ONLY written response 3c when scoring this point.** **Requirements for program code segments:** • The procedure must be student developed, but could be developed collaboratively with a partner. • If multiple procedures are included, use the first procedure to determine whether the point is earned. **Do NOT award a point if any one or more of the following is true:** • The code segment consisting of the procedure is not included in the written responses section. • The procedure is a built-in or existing procedure or language structure, such as an event handler or main method, where the student only implements the body of the procedure rather than defining the name, return type (if applicable) and parameters. • The written response describes what the procedure does independently without relating it to the overall function of the program.
Row 5 **Algorithm Implementation** **(0–1 points)** **2.B**	The written response: • includes a program code segment of a *student-developed algorithm* that includes - *sequencing* - *selection* - *iteration* • explains in detailed steps how the identified algorithm works in enough detail that someone else could recreate it.	**Consider ONLY written response 3c when scoring this point.** **Responses that do not earn the point in row 4 may still earn the point in this row.** **Requirements for program code segments:** • The algorithm being described can utilize existing language functionality or library calls. • An algorithm that contains selection and iteration, also contains sequencing. • An algorithm containing sequencing, selection, and iteration that is not contained in a procedure can earn this point. • Use the first code segment, as well as any included code for procedures called within this first code segment, to determine whether the point is earned • If this code segment calls other student-developed procedures, the procedures called from within the main procedure can be considered when evaluating whether the elements of sequencing, selection, and iteration are present as long as the code for the called procedures is included. **Do NOT award a point if any one or more of the following is true:** • The response only describes what the selected algorithm does without explaining how it does it.

Continued

		• The description of the algorithm does not match the included program code. • The code segment consisting of the selected algorithm is not included in the written response. • The algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm without explicitly identifying the code segment containing the algorithm). • The use of either the selection or the iteration is trivial and does not affect the outcome of the program.
Row 6 **Testing** **(0-1 points)** **4.C**	The written response: • describe two calls to the selected 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. • describes the condition(s) being tested by each call to the procedure. • identifies the result of each call.	**Consider ONLY the written response for 3d and the selected procedure identified in written response 3c.** **Responses that do not earn the point in row 4 may still earn the point in this row.** **Do NOT award a point if any one or more of the following is true:** • A procedure is not identified in written response 3c or the procedure does not have a parameter. • The written response for 3d does not apply to the procedure in 3c. • The two calls cause the same segment of code in the algorithm to execute even if the result is different. • The response describes conditions being tested that are implausible, inaccurate, or inconsistent with the program. • The identified results of either call are implausible, inaccurate, or inconsistent with the program.

Full Create Performance Task—Sample One

Part 1: Program Code

```
# Start Code Here
listOfStates = ["Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut", "Delaware", "Florida", "Georgia",
listOfCapitals = ["Montgomery", "Juneau", "Phoenix", "Little Rock", "Sacramento", "Denver", "Hartford", "Dover", "Tallahassee", "Atlanta",
state = input("What is your state?")
capital = input("What is your capital?")
def playStudyGame(listOfStates, listOfCapitals, state, capital):
    index = -1
    count = 0
    for x in listOfStates:
        if state == x:
            index = count
        count = count + 1
    if index == -1:
        print("Your state does not exist")
    elif listOfCapitals[index] == listOfStates[index]:
        print("You are correct")
    else:
        print("You are correct")
playStudyGame(listOfStates, listOfCapitals, state, capital)
```

Part 2: Program Video

Not shown

Part 3: Written Report

3ai. Describe the overall purpose of the program.

> New hints for the 2023 test.
>
> The most common error in the 2022 test was that students were describing how the program works, not what the user gets out of running their program.
>
> The purpose can never just be "to entertain the user." The answer needs to go beyond just entertaining the user. Example: "The purpose of my game is to entertain the user <u>and</u> develop eye-hand coordination that can be used when driving a car."
>
> To get the five points, the student must answer all three questions (3ai, 3aii, and 3aiii) correctly. No partial credit is given.
>
> Only **51%** of students got this question correct on the 2022 exam.

> The purpose of my program is to entertain the user while teaching them geography.

3aii. Describe what functionality of the program is demonstrated in the video.

New hints for the 2023 test.

The College Board wants a general description of how your program works as seen in your video. Someone who has no coding experience should be able to understand your answer. Do not go into technical details.

The program asks the user for a state and the corresponding capital. The code will find the index of where the state is in the state list, then compare the capital listed at the same index but in the capital list. If the found capital matches the user's inputted capital, the phrase "You are correct" is displayed. If the capital in the list does not match the user's chosen capital, the phrase "You are incorrect" is displayed. If the state does not exist, the phrase "Your state does not exist" is displayed.

3aiii. Describe the input and output of the program demonstrated in the video.

New hints for the 2023 test.

The College Board will play your video, pause it every time they see input or output, and check that it is in this written section. When answering this question, play your video and pause and detail all interactions.

DO NOT detail all possible outputs of your program. Only detail what is seen in your video.

Correct Answer:
The user is prompted for the state. The user **inputs** "Florida".
The user is prompted for the capital. The user inputs "Tallahassee".
The program **outputs** to the screen "You are correct".

New hints for the 2023 test.
70% of students got this question correct on the 2022 exam.
To get the five points, the student must answer all four questions (3bi, 3bii, 3biii, and 3biv) correctly. No partial credit is given.

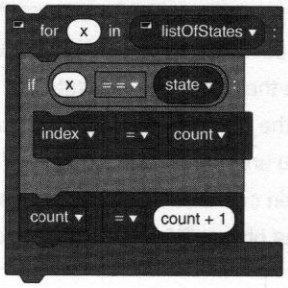
3bi. The first program code segment must show how the data have been stored in the list.

New hints for the 2023 test.
Your list must contain more than one element.

```
listOfStates ▾ = [ "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut", "Delaware", "Florida",
"Georgia", "Hawaii", "Idaho", "Illinois", "Indiana", "Iowa", "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts", "Michigan",
"Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada", "New Hampshire", "New Jersey", "New Mexico", "New York", "North Carolina",
"North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota", "Tennessee", "Texas", "Utah",
"Vermont", "Virginia", "Washington", "West Virginia", "Wisconsin", "Wyoming" ]
```

3bii. The second program code segment must show the data in the same list being used, such as creating new data from existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

> New hints for the 2023 test.
> Copy your code where you iterated through your list.



3biii. Identifies the name of the list being used in this response.

> listOfStates

3biv. Describes what the data contained in the list represent in your program.

> The data in my list are strings that represent all 50 states.

> New hints for the 2023 test.
> Answer must be specific to your program code. Use your list name and give details of your list. Do not answer this question in terms of a general list.
> Describe how the program uses your list, what changes would there be if you did not use your list, and how using your list reduced the complexity of your program.
> This question alone is worth five points. Only **36%** of students got this question correct on the 2022 test.

3bv. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list.

> By using my list **"listOfStates,"** I only had to name one variable—the list itself. If I did not use my list, I would have to create and name all **50 strings** contained in the list. My list called **listOfStates** reduced the level of complexity by reducing the amount of code I needed to write. By reducing the amount of code needed, it reduced the chances for errors.
>
> Using my list called **listOfStates** allowed me to access the strings in the list by index. If I did not use my list, I would have to access the strings by their names. Combining the index with a for loop allowed me to test each element using one if statement. If I did not use a list, I would need to use **50 if statements**. By reducing the amount of code needed, it reduced the chances for errors.

New hints for the 2023 test.

The most common error in the 2022 test was including a procedure that did not contain parameters. If the procedure does not contain parameters, the graders stop grading this question.

Iteration is your loop.

Selection is your if / if else statement

Sequence is the order of your commands. Any procedure with more than one line of code has sequence.

Only **51%** of students got this question correct on the 2022 exam.

To get the five points, the student must answer all three questions (3ci, 3cii, and 3ciii) correctly. No partial credit is given.

3ci. 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.

```
def playCapitalGame(listOfStates, listOfCapitals, state, capital):
    index = -1
    count = 0
    for x in listOfStates:
        if x == state:
            index = count
        count = count + 1
    if index == -1:
        print("Your state does not exist")
    if listOfCapitals[index] == capital:
        print("You are correct")
    if listOfCapitals[index] != capital:
        print("You are incorrect")
```

3cii. The second program code segment must show where your student-developed procedure is being called in your program.

```
playCapitalGame(listOfStates, listOfCapitals, state, capital)
```

New hints for the 2023 test.

The College Board wants a general description of how your procedure works. Not your entire program—just your procedure. Someone who has no coding experience should be able to understand your answer. Do not go into technical details.

3ciii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program.

My procedure takes in a 50-element list containing state names as well as a 50-element list containing state capital names. The user selects a state name and a capital name. The procedure then determines in what index the state variable is located in the list "listOfStates." The procedure will then check to see if that index in the second list contains the same capital name as the user-selected capital name. If they are the same, the user is correct. If they are different, the user is incorrect. If the state is not in "listOfStates," the message "Your state does not exist" is displayed.

New hints for the 2023 test.

The code should be explained line by line so that someone who can code will be able to reconstruct your procedure from your description.

The College Board uses this question to make sure the student understands their program and did not just copy code.

Only 47% of students got this question correct on the 2022 exam.

The question below is worth five points. No partial credit is given.

3civ. Describes in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

Correct Answer:

- The procedure takes in two lists, called listOfStates and listOfCapitals, as well as two strings, called state and capital.
- Sets the value of index to −1.
- Sets the value of count to 0.
- Iterates through the list "listOfStates" using a for each loop.
  - If the element of listOfStates is equal to state, the value of index is set to count.
  - Adds one to count.
- Ends the for each loop.
- If index is still set to −1, display the message "Your state does not exist."
- If the variable capital is equal to the element contained in listOfCapitals at the found index, display the message "You are correct."
- If the variable capital is not equal to the element contained in listOfCapitals at the found index, display the message "You are incorrect."

New hints for the 2023 test.
The parameters given in this example do not have to be the parameters used in the video.
The parameters must make your code run different segments of the algorithm.
To get the five points, the student must answer all three questions (3di, 3dii, and 3diii) correctly. No partial credit is given.

3di. 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:

Assume listOfStates contains "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut", "Delaware", "Florida"
and listOfCapitals contains "Montgomery", "Juneau", "Phoenix", "Little Rock", "Sacramento", "Denver", "Hartford", "Dover", "Tallahassee"
state is "Alaska"
capital is "Juneau"
Call to procedure would be: playCapitalGame (listOfStates, listOfCapitals, Alaska, Juneau)

Second call:

Assume listOfStates contains "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut", "Delaware", "Florida"
and listOfCapitals contains "Montgomery", "Juneau", "Phoenix", "Little Rock", "Sacramento", "Denver", "Hartford", "Dover", "Tallahassee"
state is "Alaska"
capital is "Dover"
Call to procedure would be: playCapitalGame (listOfStates, listOfCapitals, Alaska, Dover)

3dii. Describe what condition(s) is being tested by each call to the procedure.

Correct Answer:
Condition(s) tested by the first call:
Is the capital of Alaska Juneau?
Condition(s) tested by the second call:
Is the capital of Alaska Dover?

3diii. Identifies the result of each call

Correct Answer:
Result of the first call:
"You are correct"
Result of the second call:
"You are incorrect"

Full Create Performance Task—Sample Two

Part 1: Program Code

```
import turtle
import random
diceRollList = [] # creates empty List
def fillList(numDice):
  for i in range(numDice): # fills List
    diceRollList.append(random.randint(1,10))
def playGame(diceRollList, luckyNumber, numAppear): # procedure that has parameters
  total = 0
  for i in diceRollList: # iterate
    if i == luckyNumber: # selection
      total = total + 1
  if total == numAppear:# selection
    print("You Win")
  else: # selection
    print("You Lose")
numDice = int(input("How many dice do you want to roll? ")) # input
luckyNumber = int(input("Enter your lucky number. ")) # input
numAppear = int(input("How many times does your lucky number appear.")) # input
fillList(numDice)
playGame(diceRollList, luckyNumber, numAppear) # calls procedure
```

Part 2: Program Video

Not shown

Part 3: Written Report

3ai. Describe the overall purpose of the program.

> The purpose of my program is to entertain users while they learn a practical application of statistics.

3aii. Describe what functionality of the program is demonstrated in the video.

> The user first selects the number of dice to roll. In the video the user picks 100 dice. The dice are numbered one to ten. The user then picks their favorite number and predicts how many times the lucky number occurred in the dice rolls. In the video the user picks 5 for their lucky number and 100 for the number of times the number appears in the list. If the user guesses the number correctly, the message "You win" is displayed. If the user does not guess correctly, the message "You lose" is displayed.

3aiii. Describe the input and output of the program demonstrated in the video.

> - The user is asked how many dice they want to roll. **Input:** The user entered 1000.
> - The user is asked to enter their lucky number. **Input:** The user entered 5.
> - The user is asked how many times their lucky number appears. **Input:** The user entered 100.
> - **Output:** The phrase "You lose" is displayed.

3bi. The first program code segment must show how the data have been stored in the list.

```
diceRollList = [] # creates empty List
def fillList(numDice):
  for i in range(numDice): # fills List
    diceRollList.append(random.randint(1,10))
```

3bii. The second program code segment must show the data in the same list being used, such as creating new data from existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

```
total = 0
for i in diceRollList: # iterate
  if i == luckyNumber: # selection
    total = total + 1
```

3biii. Identifies the name of the list being used in this response.

diceRollList

3biv. Describes what the data contained in the list represent in your program.

The data in my list are random integers that represent dice rolls from 1 to 10.

3bv. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list.

My list "diceRollList" held 1000 integers. By using a list, I only had to name one variable. If I did not use a list, I would have to name all 1000 variables. By reducing the required code, I reduced the chances for errors, thus reducing the level of complexity of my program.

By using my list "diceRollList," if I wanted to change the size of the list from 1000 to 2000, no additional code would be required. If I did not use a list, I would have to name an additional 1000 variables. By reducing the required code, I reduced the chances for errors, thus reducing the level of complexity of my program.

My list "diceRollList" allowed me to access my dice rolls by index instead of by name. Using a for loop and index allowed me to test all my integers to see if they were equal to the lucky number using only one selection statement. If I did not use a list, my code would require 1000 selection statements. By reducing the required code, I reduced the chances for errors, thus reducing the level of complexity of my program.

3ci. 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.

```
7▼ def playGame(diceRollList, luckyNumber, numAppear): # procedure that has parameters
8    total = 0
9▼   for i in diceRollList: # iterate
10▼    if i == luckyNumber: # selection
11       total = total + 1
12▼  if total == numAppear:# selection
13     print("You Win")
14▼  else: # selection
15     print("You Lose")
```

3cii. The second program code segment must show where your student-developed procedure is being called in your program.

```
20  playGame(diceRollList, luckyNumber, numAppear) # calls procedure
```

3ciii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program.

My procedure playGame takes in the list of dice rolls and the user's lucky number and a guess on how many times the lucky number appears. Each dice roll is checked, and the code counts how many times the lucky number appears in the list. If the counted total is equal to the guess, the user wins. If the counted total is not equal to the guess, the user loses.

3civ. Describes in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

Line 7: My procedure playGame takes in the list diceRollList and the integer luckyNumber and the integer numAppear.
Line 8: Sets the counter variable total to 0.
Line 9–11: Uses a for each loop to iterate through the list. If the value in the list is equal to luckyNumber, add one to total.
Line 12: If total is equal to numAppear output to the screen, "You Win."
Line 13: If total is not equal to numAppear output to the screen, "You Lose."

3di. 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:

Assuming diceRollList contained the numbers [1, 1, 3, 5, 6] and lucky number was 1 and numAppear was 2.
playGame(diceRollList, luckyNumber, numAppear) would be playGame([1, 1, 3, 5, 6], 1, 2]

Second call:

Assuming diceRollList contained the numbers [1, 1, 3, 5, 6] and lucky number was 1 and numAppear was 3.
playGame(diceRollList, luckyNumber, numAppear) would be playGame([1, 1, 3, 5, 6], 1, 3]

3dii. Describe what condition(s) is being tested by each call to the procedure.

Condition(s) tested by the first call:

The procedure is testing if the number 1 is contained two times in the list.

Condition(s) tested by the second call:

The procedure is testing if the number 1 is contained three times in the list.

3diii. Identifies the result of each call

Result of the first call:

"You win"

Result of the second call:

"You lose"

Full Create Performance Task—Sample Three

Part 1: Program Code

```
import random
listSize = int(input("How many numbers do you want to play with? "))
listOfNumbers = []

def fillList (listSize):
  for x in range(listSize):
    listOfNumbers.append(random.randint(0, 10))

def playGame (listOfNumbers, guess):
  total = 0
  for item in listOfNumbers:
    if item == guess:
      listOfNumbers.remove(item)
      total = total + 1
  if ( total == 0):
    return -1
  else:
    return total

fillList(listSize)
play = 1
score = 0
while play > 0:
  pick = int(input("pick a number "))
  play = (playGame(listOfNumbers,pick ))
  print(play)
  if play == -1:
    print ("Game Over.")
    print ("Your score is " + str(score))
  else:
    score = score + 10
```

Part 2: Program Video

Not shown

Part 3: Written Report

3ai. Describe the overall purpose of the program.

> Correct Answer:
>
> The purpose of my program is to entertain the user while teaching them statistics.

3aii. Describe what functionality of the program is demonstrated in the video.

> The program asks the user how many numbers they want to play with. The user then guesses a number. If that number is contained in the list, the number is removed from the list and the user scores 10 points. If the user guesses a number that is not contained in the list, the game is over and the total points won is displayed. The game continues until the user guess a number not contained in the list.

3aiii. Describe the input and output of the program demonstrated in the video.

> The user is prompted to enter how many numbers they want to play with. The user **inputs** 16.
>
> The user is prompted to pick a number. The user **inputs** 3. The program **outputs** to the screen 2.
>
> The user is prompted to pick a number. The user **inputs** 6. The program **outputs** to the screen 3.
>
> The user is prompted to pick a number. The user **inputs** 6. The program **outputs** to the screen the following...
>
> –1
>
> Game Over.
>
> Your score is 20

3bi. The first program code segment must show how the data have been stored in the list.

```
2   listSize = int(input("How many numbers do you want to play with? "))
3   listOfNumbers = []
4
5 ▼ def fillList (listSize):
6 ▼   for x in range(listSize):
7       listOfNumbers.append(random.randint(0, 10))
```

3bii. The second program code segment must show the data in the same list being used, such as creating new data from existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

```
10    total = 0
11▼   for item in listOfNumbers:
12▼     if item == guess:
13        listOfNumbers.remove(item)
14        total = total + 1
```

3biii. Identifies the name of the list being used in this response.

listOfNumbers

3biv. Describes what the data contained in the list represent in your program.

The data in my list are random integers from 0 to 10 that represent the numbers used for the guessing game.

3bv. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list.

By using my list "**listOfNumbers**," I only had to name one variable—the list itself. If I did not use my list, I would have to create and name all **16 integers** contained in the list. My list called **listOfNumbers** reduced the level of complexity by reducing the amount of code I needed to write. By reducing the amount of code needed, it reduced the chances for errors.

Using my list called **listOfNumbers** allowed me to access the numbers in the list by index. If I did not use the list, I would have to access the numbers by their names. Combining the index with a for loop allowed me to test each element using one if statement. If I did not use a list, I would need to use **16 if statements**. By reducing the amount of code needed, it reduced the chances for errors.

By using my list "**listOfNumbers**," if I wanted to change the size of the list from 16 to 2000, no additional code would be required. If I did not use a list, I would have to name an additional 1984 variables. By reducing the required code, I reduced the chances for errors, thus reducing the level of complexity of my program.

3ci. 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.

```
9▼ def playGame (listOfNumbers, guess):
10     total = 0
11▼    for item in listOfNumbers:
12▼       if item == guess:
13           listOfNumbers.remove(item)
14           total = total + 1
15▼    if ( total == 0):
16        return -1
17▼    else:
18        return total
```

3cii. The second program code segment must show where your student-developed procedure is being called in your program.

```
25  play = (playGame(listOfNumbers,pick ))
```

3ciii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program.

My procedure takes in a list of random numbers and a number guessed by the user. The procedure returns the number of times the number appears in the list. If the number is not found, −1 is returned. All occurrences of the guessed number are cleared from the list after being counted.

3civ. Describes in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

Line 9: The procedure playGame takes in a list of numbers and a number guessed by the user.

Line 10: The variable total is set to 0.

Line 11–14: The list listOfNumbers is iterated and if the guessed number is in the list, it is removed and one is added to total.

Line 15–16: If total is still equal to 0, −1 is returned.

Line 17–18: In all other cases, the total number of times the guessed number was found in the list is returned.

3di. 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:

Assume listOfNumbers contains [1, 1, 3, 5, 6, 7] and guess is 1.

The call would be playGame(listOfNumbers, 1).

Second call:

Assume listOfNumbers contains [1, 1, 3, 5, 6, 7] and guess is 4.

The call would be playGame(listOfNumbers, 4).

3dii. Describe what condition(s) is being tested by each call to the procedure.

Condition(s) tested by the first call:

Is 1 contained in the list if yes how many times.

Condition(s) tested by the second call:

Is 4 contained in the list if yes how many times.

3diii. Identifies the result of each call

> Result of the first call:
>
> The number 2 is returned.
>
> Result of the second call:
>
> The number −1 is returned.

Full Create Performance Task—Sample Four

Part 1: Program Code

```
keyList = [1, 4, 6, 9, 5, 7, 3, 1, 6, 9, 2, 3, 4, 5, 6, 1, 2, 8, 7, 6, 5, 4, 6, 5,
           2, 4, 5, 1, 7, 8, 5, 4, 3, 2, 4, 5, 6, 7, 8, 2, 5, 3, 9, 1, 0, 3]

def scrambleWord(word, enOrDe):
  tempList = []
  for i in word:
    tempList.append(i)
  result = ""
  x = 0
  for item in keyList:
    if x < len(word):
      if (item % 2 == 0):
        if enOrDe == "e":
          result = result + chr((ord(tempList[x] )+ item))
        else:
          result = result + chr((ord(tempList[x] )- item))
      else:
        if enOrDe == "e":
          result = result + chr((ord(tempList[x])+ item - 2))
        else:
          result = result + chr((ord(tempList[x])- item + 2))
    x = x + 1
  print (result)


word = (input("Enter your phrase: "))
enOrDe = (input("Do you want to encrypt or decrypt your message? (e/d) "))
scrambleWord(word, enOrDe)
```

Part 2: Program Video

Not shown

Part 3: Written Report

3ai. Describe the overall purpose of the program.

> The purpose of my program is to allow the user to scramble messages to increase privacy when sending communications.

3aii. Describe what functionality of the program is demonstrated in the video.

> The program asks the user to enter a phrase. If the user chooses to encrypt the phrase, the phrase will be scrambled based on the key contained in the list keyList. If the key number is even, the program will add the key value to the character. If the key value is odd, the program will add the key value plus 2. If the user chooses to decrypt the phrase, the phrase will be unscrambled based on the key contained in the list keyList. If the key number is even, the program will subtract the key value from the character. If the key value is odd, the program will subtract the key value minus 2.

3aiii. Describe the input and output of the program demonstrated in the video.

> The user is prompted the enter their phrase. The user **inputs** "Dogs like socks but not on their feet."
>
> The user is then prompted to indicate if they want the program to encrypt or decrypt. The user **inputs** "e"
>
> The program then **outputs** Csm.v%mhql"tsfqr"jzz#ruw"sqyphms"jhky6

3bi. The first program code segment must show how the data have been stored in the list.

```
keyList = [1, 4, 6, 9, 5, 7, 3, 1, 6, 9, 2, 3, 4, 5, 6, 1, 2, 8, 7, 6, 5, 4, 6, 5,
           2, 4, 5, 1, 7, 8, 5, 4, 3, 2, 4, 5, 6, 7, 8, 2, 5, 3, 9, 1, 0, 3]
```

3bii. The second program code segment must show the data in the same list being used, such as creating new data from existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

```
for item in keyList:
    if x < len(word):
      if (item % 2 == 0):
        if enOrDe == "e":
          result = result + chr((ord(tempList[x] )+ item))
        else:
          result = result + chr((ord(tempList[x] )- item))
      else:
        if enOrDe == "e":
          result = result + chr((ord(tempList[x])+ item - 2))
        else:
          result = result + chr((ord(tempList[x])- item + 2))
    x = x + 1
```

3biii. Identifies the name of the list being used in this response.

> keyList

3biv. Describes what the data contained in the list represent in your program.

> The data in my list are integers that represent how much each character in the user's phrase will be shifted.

3bv. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list.

> By using my list **"keyList,"** I only had to name one variable—the list itself. If I did not use my list, I would have to create and name all **46 integers** that are contained in the list. "keyList" reduced the level of complexity by reducing the amount of code I needed to write. By reducing the amount of code needed, it reduced the chances for errors.
>
> Using my list called **"keyList"** allowed me to access the integers in the list by index. If I did not use my list, I would have to access the integers by their names. Combining the index with a for loop allowed me to test each element using one if statement. If I did not use a list, I would need to use **46 separate if statements**. By reducing the amount of code needed, it reduced the chances for errors.

3ci. 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.

```
4▼ def scrambleWord(word, enOrDe):
5     tempList = []
6▼    for i in word:
7        tempList.append(i)
8     result = ""
9     x = 0
10▼   for item in keyList:
11▼      if x < len(word):
12▼        if (item % 2 == 0):
13▼          if enOrDe == "e":
14             result = result + chr((ord(tempList[x] )+ item))
15▼          else:
16             result = result + chr((ord(tempList[x] )- item))
17▼        else:
18▼          if enOrDe == "e":
19             result = result + chr((ord(tempList[x])+ item - 2))
20▼          else:
21             result = result + chr((ord(tempList[x])- item + 2))
22       x = x + 1
23    print (result)
```

3cii. The second program code segment must show where your student-developed procedure is being called in your program.

```
28 scrambleWord(word, enOrDe)
```

3ciii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program.

My procedure takes in a 46-element list containing integers as well as a string that indicates if the phrase will be encrypted or decrypted. Each letter in the phrase matches with a corresponding key number. If the key number is even and encryption is selected, the letter is moved forward equal to the key number. If the key number is odd and encryption is selected, the letter is moved forward equal to the key number minus two. If the key number is even and decryption is selected, the letter is moved backward equal to the key number. If the key number is odd and decryption is selected, the letter is moved backward equal to the key number plus two. The new word is then displayed on the screen.

3civ. Describes in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

Line 4: The procedure scrambleWord takes in a phrase called "word" and a string called "enOrDe".

Line 5: The empty list tempList is created.

Line 6–7: A for loop is executed the same size as the length of word and appends each letter the word to a character into the list tempList.

Line 8: A empty string called "result" is created.

Line 9: x is set to 0.

Line 10: A for loop is set up to iterate through the list keyList.

Line 11: If the value of x is less then the length of word and...

Line 12: If the current element of keyList is even and...

Line 13–14: If enOrDe is set to encrypt shift the character forward the keyList value and is added to result.

Line 15–16: Else if enOrDe is set to decrypt shift the character backwards the keyList value and is added to result.

Line 17: Else if the current element of keyList is odd and...

Line 18–19: If enOrDe is set to encrypt shift the character forward the keyList value - 2 and is added to result.

Line 20–21: Else if enOrDe is set to decrypt shift the character backwards the keyList value + 2 and is added to result.

Line 22: One is added to x.

Line 23: Go to line 11 until the for loop is complete.

Line 23: Displays the value of the translated word in result.

3di. 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:

```
scrambleWord( "Mississippi", e)
```

Second call:

```
scrambleWord("Lmyzlxthvwk", d)
```

3dii. Describe what condition(s) is being tested by each call to the procedure.

Condition(s) tested by the first call:

Encrypt the word "Mississippi" based on the stored key.

Condition(s) tested by the second call:

Decrypt the word "Lmyzlxthvwk" based on the stored key.

3diii. Identifies the result of each call

Result of the first call:

Lmyzlxthvwk

Result of the second call:

Mississippi

Full Create Performance Task—Sample Five

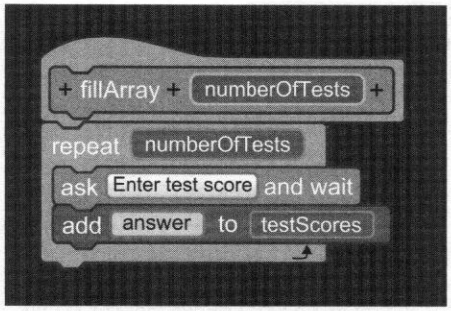
Graphical-Based Language

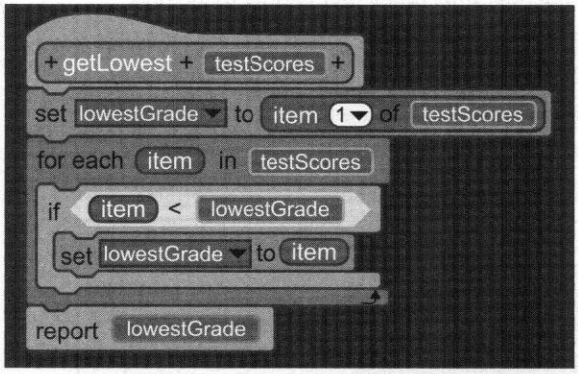
This program can be used to calculate averages on tests. The program can calculate the average or can calculate the average with the lowest test score dropped.

Part 1: Program Code

```
when [flag] clicked
ask [How many tests] and wait
set [numberOfTests] to [answer]
set [testScores] to [list]
fillArray [numberOfTests]
set [min] to [getLowest] [testScores]
say [findAverage] [testScores] [true] for [20] secs
say [findAverage] [testScores] [false] for [20] secs
```

```
+findAverage + [testScores] + [dropLowest] +
set [total] to [0]
if [dropLowest] = [true]
  set [count] to [0]
  repeat [numberOfTests]
    change [count] by [1]
    set [total] to [total] + [item] [count] [of] [testScores]
  report [total] - [min] / [count] - [1]
else
  set [count] to [0]
  repeat [numberOfTests]
    change [count] by [1]
    set [total] to [total] + [item] [count] [of] [testScores]
  report [total] / [count]
```





Part 2: Program Video

Not shown

Part 3: Written Report

3a. You can write approximately 150 words for all subparts of 3a combined.

Purpose: The problem being solved or creative interest being pursued through the program.

i. Describe the overall purpose of the program.

The purpose of my program is to allow teachers to quickly calculate student averages with and without the lowest grade dropped. By saving time, teachers will have additional class time to cover additional content.

Functionality: The behavior of a program during execution, often described by how the user interacts with it.

ii. Describe what functionality of the program is demonstrated in the video.

The program **functions** by asking the user how many test scores to include and then fills up the list at that size by asking the user to input all the student's grades in a loop. After the list is filled, the program asks the user if he or she wants to calculate the average with or without the lowest test score dropped. The minimum number in the list is found, and the average is determined with or without the minimum number.

Input: Program input is data that are sent to a computer for processing by a program. Input can come in a variety of forms, such as tactile, audible, visual, or text. An event is associated with an action and supplies input data to a program.

Output: Program output is any data that are sent from a program to a device. Program output can come in a variety of forms, such as tactile, audible, visual, or text.

iii. Describe the input and output of the program demonstrated in the video.

The user is asked "How many tests". The user inputs 4.
The user is asked "Enter test score".
The user inputs 90.
The user is asked "Enter test score".
The user inputs 100.
The user is asked "Enter test score".
The user inputs 60.
The user is asked "Enter test score".
The user inputs 80.
The program outputs to the screen "90" and "82.5".

3b. Capture and paste two program segments you developed during the administration of this task that contain a list (or other collection types, including arrays) being used to manage complexity in your program.

You can write approximately 200 words for all subparts of 3b combined, exclusive of program code.

Data have been stored in the list. Input into the list can be through an initialization or through some computation on other variables or list elements.

List: An ordered sequence of elements. The use of lists allows multiple related items to be represented using a single variable. Lists are referred to by different terms, such as arrays or arrayLists, depending on the programming language.

i. The first program code segment must show how data has been stored in the list.

```
+ fillArray + [numberOfTests] +
repeat [numberOfTests]
    ask [Enter test score] and wait
    add [answer] to [testScores]
```

ii. The second program code segment must show the data in the same list being used, such as creating new data from existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

```
+ getLowest + [testScores] +
set [lowestGrade] to [item (1) of testScores]
for each [item] in [testScores]
    if [item < lowestGrade]
        set [lowestGrade] to [item]
report [lowestGrade]
```

Then provide a written response that does all three of the following:

iii. Identifies the name of the variable representing the list being used in this response.

The name of the list is testScores.

iv. Describes what the data contained in the list are representing in the program.

The data contained in the testScores are integers that represent a student's test scores. These test scores are used to calculate the average test score or the average test score with the lowest score dropped as well as to determine the lowest test score.

v. Explains how the named, selected list manages complexity in the program code by explaining why the program code could not be written, or how it would be written differently, without using the list.

My list testScores holds four test scores. By using my list I only had to name the list. If I did not use testScores I would have to name all four of the tests with their own variable names. This reduction in required code will reduce the chances for error. By reducing the chances of errors it reduced the level of complexity of my program.
By putting my test scores in testScores I was able to reference and test each score by index using a loop and a single if statement. If I did not use my list I would have to reference each score by its name and use four different if statements. This reduction in required code will reduce the chances for error. By reducing the chances of errors it reduced the level of complexity of my program.
Furthermore, by using my list I could store four scores or 100 scores without an additional code. If I did not use my list I would have to name an additional 96 scores. This reduction in required code will reduce the chances for error. By reducing the chances of errors it reduced the level of complexity of my program.

3c. Include 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. You can use approximately 200 words for all subparts of 3c combined, not including program code.
WARNING: Built-in or existing procedure and language structures, such as event handlers and main methods, are NOT considered student developed.

Student-developed procedure/algorithm: Program code that is student developed has been written (individually or collaboratively) by the student who submitted the response. Calls to existing program code or libraries can be included but are not considered student developed. Event handlers are built-in abstractions in some languages and are therefore not considered student developed. In some block-based programming languages, event handlers begin with "when."

Procedure: A named group of programming instructions that may have parameters and return values. Procedures are referred to by different names, such as method or function, depending on the programming language.

i. The first program 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 influence the functionality of the procedure.
   - Implements an algorithm that includes sequencing, selection, and iteration.

Parameter: An input variable of a procedure.
Algorithm: A finite set of instructions that accomplish a specific task. Every algorithm can be constructed using a combination of sequencing, selection, and iteration.
Sequencing: The application of each step of an algorithm in the order in which the code statements are given.
Iteration: A repetitive portion of an algorithm. Iteration repeats until a given condition is met or for a specified number of times.

```
+ findAverage + testScores + dropLowest +
set total to 0
if dropLowest = true
    set count to 0
    repeat numberOfTests
        change count by 1
        set total to total + item count of testScores
    report total - min / count - 1
else
    set count to 0
    repeat numberOfTests
        change count by 1
        set total to total + item count of testScores
    report total / count
```

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

```
say findAverage testScores true for 20 secs
say findAverage testScores false for 20 secs
```

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 included procedure uses the user's input to calculate average. If the user enters "true" the lowest test grade is dropped from the average. If the user enters "false" the lowest grade is included in the average. After calculation the average is returned as output.

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 procedure first checks the value of a user-entered input (true/false).
If the user enters "true", the procedure iterates through the data structure and accumulates the total value of the test scores. This program starts accumulating from the first to the last indexed test stored in the data structure.
The procedure also keeps track of how many tests are in the data structure by adding 1 to count for every iteration.
The procedure then returns the (total score – the lowest test score)/(the number of tests – 1).
If the user enters "false", the procedure iterates through the data structure and accumulates the total value of the test scores. This program starts accumulating from the first to the last indexed test stored in the data structure.
The procedure also keeps track of how many tests are in the data structure by adding 1 to count for every iteration.
The procedure then returns the (total score)/(the number of tests).

3d. Provide a written response that does all three of the following (using approximately 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.

Argument(s): The value(s) of the parameter(s) when a procedure is called.

First call:

```
say findAverage testScores true for 20 secs
```

The first call to the procedure includes two arguments. The first argument is the list that contains the grades. The second argument is a Boolean value that indicates that the lowest test score should be dropped. The selection statement in the procedure executes as true, and the average is calculated with the lowest score dropped.

Second call:

```
say findAverage testScores false for 20 secs
```

The second call to the procedure includes two arguments. The first argument is the list that contains the grades. The second argument is a Boolean value that indicates that the lowest test score should not be dropped.

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

The selection statement in the procedure executes as "true," and the average is calculated with the lowest score dropped.

Condition(s) tested by the second call:

The selection statement in the procedure executes as "false," and the average is calculated with all scores from the data structure.

iii. Identifies the result of each call.

Result of first call:

95

Result of second call:

90

Full Create Performance Task—Sample Six

Text Based—Easy!

Part 1: Program Code

```
import java.util.Scanner // imports the scanner class
public class Average;
{
  public static void main(String args[])
  {
    Scanner scan = new Scanner(System.in);
    System.out.println("How many tests?");
    int numTests = scan.nextInt(); // reads input
    //Calling the fillyArray procedure
    int[] testScores = fillyArray(numTests);
    System.out.println("Drop lowest grade? (y/n) ");
    char dropLowest = scan.next().charAt(0); // reads input
    //Calling the getAverage procedure
    double average = getAverage(testScores, dropLowest);
    System.out.println("The average is: " + average);
  }
```

```
17 public static int[] fillArray(int num)
18 {
19   int[] tempTestScores = new int[num];
20   Scanner scan = new Scanner(System.in); // Scanner Libary
21   for(int x = 0 ; x < tempTestScores.length; x++)
22   {
23     System.out.print("Enter Test Value: ");
24     tempTestScores[x] = scan.nextInt();
25   }
26   return tempTestScores;
27 }
28 public static double getAverage(int[] testScores, char dropLowest)
29 {
30   int count = 0;
31   int total = 0;
32   for(int x = 0 ; x < testScores.length, x++)
33   {
34     total = total + testScores[x];
35     count = count + 1;
36   }
37   if(dropLowest == 'y')
38   {
39     int min = testScores[0];
40     for(int x = 0; x < testScores.length; x++)
41     {
42       if(testScores[x] < min)
43         min = testScores[x];
44     }
45     return (total - min) / (count - 1);
46   }
47   else
48       return total / count;
49   }
50 }
```

Part 2: Program Video

Not shown

Part 3: Written Responses

3a. You can write approximately 150 words for all subparts of 3a combined.

Purpose: The problem being solved or creative interest being pursued through the program.

i. Describe the overall purpose of the program.

The purpose of my program is to allow teachers to quickly calculate student averages with and without the lowest grade dropped. By saving time, teachers will have additional class time to cover additional content.

Functionality: The behavior of a program during execution and is often described by how the user interacts with it.

ii. Describe what functionality of the program is demonstrated in the video.

The program **functions** by asking the user how many test scores to include and then fills up the list at that size by asking the user to input all the student's grades in a loop. After the list is filled, the program asks the user if he or she wants to calculate the average with or without the lowest test score dropped. The minimum number in the list is found, and the average is determined with or without the minimum number.

Input: Program input is data that are sent to a computer for processing by a program. Input can come in a variety of forms, such as tactile, audible, visual, or text. An event is associated with an action and supplies input data to a program.

Output: Program output is any data that are sent from a program to a device. Program output can come in a variety of forms, such as tactile, audible, visual, or text.

iii. Describe the input and output of the program demonstrated in the video

The user is asked "How many tests". The user inputs 4.
The user is asked "Enter test score".
The user inputs 90.
The user is asked "Enter test score".
The user inputs 100.
The user is asked "Enter test score".
The user inputs 60.
The user is asked "Enter test score".
The user inputs 80.
The program outputs to the screen "90" and "82.5".

3b. Capture and paste two program segments you developed during the administration of this task that contain a list (or other collection including arrays) being used to manage complexity in your program.

You can write approximately 200 words for all subparts of 3b combined, exclusive of program code.

List: An ordered sequence of elements. The use of lists allows multiple related items to be represented using a single variable. Lists are referred to by different terms, such as arrays or arrayLists, depending on the programming language.

Data have been stored in the list: Input into the list can be through an initialization or through some computation on other variables or list elements.

i. The first program code segment must show how data have been stored in the list.

```
public static int[] fillArray(int num)
{
  int[] tempTestScores = new int[num];
  Scanner scan = new Scanner(System.in); // Scanner Libary
  for(int x = 0 ; x < tempTestScores.length; x++)
  {
    System.out.print("Enter Test Value: ");
    tempTestScores[x] = scan.nextInt();
  }
  return tempTestScores;
}
```

ii. The second program code segment must show the data in the same list being used, such as creating new data from existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

```
public static double getAverage(int[] testScores, char dropLowest)
{
  int count = 0;
  int total = 0;
  for(int x = 0 ; x < testScores.length, x++)
  {
    total = total + testScores[x];
    count = count + 1;
  }
  if(dropLowest == 'y')
  {
    int min = testScores[0];
    for(int x = 0 ; x < testScores.length; x++)
    {
      if(testScores[x] < min)
        min = testScores[x];
    }
    return (total - min) / (count - 1);
  }
  else
    return total / count;
}
```

Then provide a written response that does all three of the following:

iii. Identifies the name of the variable representing the list being used in this response.

The name of the list is testScores.

iv. Describes what the data contained in the list are representing in the program.

The data contained in the testScores are integers that represent a student's test scores. These test scores are used to calculate the average test score or the average test score with the lowest score dropped as well as to determine the lowest test score.

v. Explains how the named, selected list manages complexity in the program code by explaining why the program code could not be written, or how it would be written differently, without using the list.

My list testScores holds four test scores. By using my list I only had to name the list. If I did not use testScores I would have to name all four of the tests with their own variable names. This reduction in required code will reduce the chances for error. By reducing the chances of errors it reduced the level of complexity of my program.
By putting my test scores in testScores I was able to reference and test each score by index using a loop and a single if statement. If I did not use my list I would have to reference each score by their name and use four different if statements. This reduction in required code will reduce the chances for error. By reducing the chances of errors it reduced the level of complexity of my program. Furthermore, by using my list I could store four scores or 100 scores without an additional code. If I did not use my list I would have to name an additional 96 scores. This reduction in required code will reduce the chances for error. By reducing the chances of errors it reduced the level of complexity of my program.

3c. Include 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. You can use approximately 200 words for all subparts of 3c combined, not including program code.
WARNING: Built-in or existing procedure and language structures, such as event handlers and main methods, are NOT considered student developed.

Student-developed procedure/algorithm: Program code that is student developed has been written (individually or collaboratively) by the student who submitted the response. Calls to existing program code or libraries can be included but are not considered student developed. Event handlers are built-in abstractions in some languages and are therefore not considered student developed. In some block-based programming languages, event handlers begin with "when."

Procedure: A named group of programming instructions that may have parameters and return values. Procedures are referred to by different names, such as method or function, depending on the programming language.

i. The first program 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 influence the functionality of the procedure.
   - Implements an algorithm that includes sequencing, selection, and iteration.

Parameter: An input variable of a procedure.
Algorithm: A finite set of instructions that accomplish a specific task. Every algorithm can be constructed using a combination of sequencing, selection, and iteration.
Sequencing: The application of each step of an algorithm in the order in which the code statements are given.
Iteration: A repetitive portion of an algorithm. Iteration repeats until a given condition is met or for a specified number of times.

```
public static double getAverage(int[] testScores, char dropLowest)
{
  int count = 0;
  int total = 0;
  for(int x = 0 ; x < testScores.length, x++)
  {
    total = total + testScores[x];
    count = count + 1;
  }
  if(dropLowest == 'y')
  {
    int min = testScores[0];
    for(int x = 0 ; x < testScores.length; x++)
    {
      if(testScores[x] < min)
        min = testScores[x];
    }
    return (total - min) / (count - 1);
  }
  else
     return total / count;
}
```

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

Line 10 and Line 14

```
 8 int numTests = scan.nextInt(); //reads input
 9 //Calling the fillyArray procedure
10  int[] testScores = fillyArray(numTests);
11  System.out.println["Drop lowest grade? (y/n)");
12  char dropLowest = scan.next() .charAt(0);//reads input
13  //Calling the getAverage procedure
14  double average = getAverage(testScores, dropLowest);
15  System.out.println("The average is: " + average);
```

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 included procedure uses the user's input to calculate average. If the user enters "true" the lowest test grade is dropped from the average. If the user enters "false" the lowest grade is included in the average. After calculation the average is returned as output.

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 procedure first checks the value of a user-entered input (true/false).

If the user enters "true", the procedure iterates through the data structure and accumulates the total value of the test scores. This program starts accumulating from the first to the last indexed test stored in the data structure.

The procedure also keeps track of how many tests are in the data structure by adding 1 to count for every iteration.

The procedure then returns the (total score – the lowest test score)/(the number of tests – 1).

If the user enters "false", the procedure iterates through the data structure and accumulates the total value of the test scores. This program starts accumulating from the first to the last indexed test stored in the data structure.

The procedure also keeps track of how many tests are in the data structure by adding 1 to count for every iteration.

The procedure then returns the (total score)/(the number of tests).

3d. Provide a written response that does all three of the following (using approximately 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.

Argument(s): The value(s) of the parameter(s) when a procedure is called.

First call:

double average = getAverage(testScores, 'y'); The first call to the procedure includes two arguments. The first argument is the list that contains the grades. The second argument is a char value that indicates that the lowest test score should be dropped. The selection statement in the procedure executes as true, and the average is calculated with the lowest score dropped.

Second call:

double averageFull = getAverage(testScores, 'n'); The second call to the procedure includes two arguments. The first argument is the list that contains the grades. The second argument is a char value that indicates that the lowest test score will not be dropped.

ii. Condition(s) tested by the first call:

The selection statement in the procedure executes as "true," and the average is calculated with the lowest score dropped.

Condition(s) tested by the second call:

The selection statement in the procedure executes as "false," and the average is calculated with all scores from the data structure.

iii. Identifies the result of each call.
Result of first call:

95

Result of second call:

90

Create Performance Task Seven

Program code only.

This sample program contains all the features needed to cover the AP create rubric using JAVA.

```
import java.util.*; // imports scanner libary
                    // NOT student developed
public class Cypher
{
  public static void main(String args[])
  {
    Scanner scan = new Scanner(System.in); //creates Scanner object
    System.out.println("Enter word to be encrypted");
    String word = scan.nextLine(); // scanner method

     System.out.println("Enter key length");
     int keyLength = scan.nextlnt(); // scanner method

     // ************ calls student developed method ************
     char[] chacterArray = fillArray(word);
     // ************ calls student developed method ************
     int[] key = getKey(keyLength);
     // ************ calls student developed method ************
     char[] scramble = changeWord(chacterArray,key, true);
     // ************ calls student developed method ************
     char[] deScramble = changeword(scramble,key, false);

   }
   // ************ student developed method ************
   public static char[] changeWord (char[] cArray, int[] key, boolean en )
   {
     char [] temp = new char [cArray.length];
     if (en == false )
     {
       System.out.print("Actual word: ");
       for (int x = 0 ; x < cArray.length; x++)
       {
           temp [x] = (char) (cArray[x] - key [x % key.length]);
           System.out.print(temp[x] + "");
       }
       System.out.println();
       return temp;
     }

```

```
    else
    {
      System.out.print("Scrambled word: ");
      for(int x = 0; x < cArray.length; x++)
      {
        temp[x] = (char)(cArray[x] + key[x % key.length]);
        System.out.print((char)(temp[x]) + "");
      }
      System.out.println();
      return temp;
    }
  }

  // ************ student developed method ************
  public static char[] fillArray(String temp)
  {
    char[] t = new char[temp.length()];
    for(int x = 0 ; x < t.length ; x++)
    {
      t[x] = temp.charAt(x);
    }
    return t;
  }

  // ************ student developed method ************
  public static int[] getKey(int keyLength)
  {
      Scanner scan = new Scanner(System.in);
      int[] key = new int[keyLength];
      for(int x = 0 ; x < key.length ; x++)
      {
        key[x] = scan.nextInt();
      }
      return key;
  }
}
```

Create Performance Task Eight

High Low Dice Game—Medium Difficulty but Fun

Program code only.

This sample program contains all the features needed to cover the AP create rubric using SNAP!

```
when [flag] clicked
set randomNumbers to list
set dollars to 0
set continue to 0
set high to 0
set low to 0
set highMatch to 0
set lowMatch to 0
set bel to 0
ask How many dice rolls? and wait
set rolls to answer
fillArray rolls
repeat until continue = -1
    ask Enter high match and wait
    set highMatch to answer
    ask Enter low match and wait
    set lowMatch to answer
    ask How many dollars do you want to bet? and wait
    set bel to answer
    set high to checkForMatch highMatch
    set low to checkForMatch lowMatch
    if high > low
        set dollars to dollars + bet
    if low > high
        set dollars to dollars - bet
    ask Enter-1 to end game or any other number to continue. and wait
    set continue to answer
say dollars
```

```
+ fillArray + rolls +
repeat rolls
    add pick random 1 to 6 to randomNumbers
```

Create Performance Task Nine

Math Help—Easy

Program code only.

This sample program contains all the features needed to cover the AP create rubric using JAVA.

```
1   // Imports the Scanner class to use the scanner methods
2   import java.util.*;
3   public class MathHelper
4   {
5     public static void main(String args[])
6     {
7       // Calls to a students created procedure
8       int[] numbersOne = fillArrayWithNumbers();
9       // Calls  to a students created procedure
10      int[] numbersTwo = fillArrayWithNumbers();
11      // Calls to a students created procedure
12      giveTest(numbersOne,numbersTwo, "add");
13      // Calls to a students created procedure
14      giveTest(numbersOne,numbersTwo, "sub");
15    }
16
17  // A student created procedure that takes in 2 integer arrays and a
    string
18  public static void giveTest (int[] numsOne, int[] numsTwo, String type)
19  {
20    int answer = 0;
21    int playerAnswer = 0;
22    int score = 0;
23    int count = 0;
24    Scanner scan = new Scanner(System.in);
25    if(type.equals("add"))
26    {
27      for(int x = 0 ; x < numsOne.length ; x++)
28      {
29        answer = numsOne[x] + numsTwo[x];
30        System.out.println( numsOne[x] + " + " + numsTwo[x] + " = ");
31        System.out.print("enter answer");
32        playerAnswer = scan.nextInt();
33        count = count + 1;
34        if( playerAnswer == answer)
```

```
35              {
36                System.out.println("Correct!!!");
37                score = score + 1;
38                }
39                else
40                  System.out.println("Sorry incorrect!!!");
41            }
42              // Outputs score in %
43              System.out.println("Your score is " +(int)(1.0 * score / count *
                100));
44          }
45          else if(type.equals("sub"))
46          {
47            for(int x = 0 ; x < numsOne.length ; x++)
48            {
49              answer = numsOne[x] - numsTwo[x];
50              System.out.println( numsOne[x] + " - " + numsTwo[x] + " = ");
51              System.out.print("enter answer");
52              playerAnswer = scan.nextInt();
53              count = count + 1;
54              if( playerAnswer == answer)
55              {
56                System.out.println("Correct!!!");
57                score = score + 1;
58              }
59                else
60                  System.out.println("Sorry incorrect!!!");
61                System.out.println("Your score is " +(int)(1.0 * score / count *
                  100));
62
63              }
64            }
65          }
66
67
68          // user created procedure that fills the array with
69          // integers 1, 2 or 3 randomly
70          public static int[] fillArrayWithNumbers()
71          {
72            int[] nums = new Int[3];
73            for(int x = 0 ; x < nums.length ; x++)
74            {
75              nums[x] = (int) (Math.random() * 100) + 1;
76            }
77            return nums;
78
79          }
80        }
```