Your unit plan draft proposal should have the following parts: * High level description of unit content. * Rationale for creating the unit. * Placement of unit in existing class/sequence.

Unit Plan Draft Proposal

Name: Unit 2 - Abstractions

Contributors: Victoria Berkowitz and Stephannia Kabanakis

Language: Hatch! (Block-based coding)

Description:

This unit covers the following topics:

- Abstraction
- Introduction to Lists (What is a list? List methods and indices)
- List Programming
- Algorithms (What is an algorithm and where do we use them in our every day lives?)

Rationale:

- We created this unit because we feel that abstraction pairs well with lists.
- We also added in algorithms as we get more into the idea that computers take specific instructions and being clear and direct is a neccessity.
- After learning strings, students will have many more opportunities to practice programming and interact with the previous lesson of variables as well.
- Including algorithms will help better prepare students with the concepts of procedures and help tie this unit into the next.
- We have removed strings from this unit because we plan to have it covered in unit 1 by having students practice with displaying variables and string concatenation.

Progression:

- Abstraction: We will go over abstraction, what it is, and how it is used in our everyday lives. We plan on using the names example (how names for people are abstractions) and tie this into the idea of variables as abstractions. This lesson on abstraction will also cover dynamic and static project vocabulary as well as tie in these concepts to the Create Performance Task portions of the APCSP Exam. We plan on giving students their first taste of the CPT by giving them already working code and having them answer questions about the code similar to the CPT, but focused on variables(because they will be finishing their unit on variables the lesson before this.)
- Introduction to Lists: We plan to go over lists in everyday lives, review simple list vocab(list, element, index, length). We then will cover different list methods in Hatch! and what they look like. After, we plan to give an offline activity where we create a wish list as a class. Then we will ask students to alter the list based on a set of instructions. We will review each instruction and model to students how the changes occured.
- List Programming: We will give students 2-3 coding challenges on lists to help them practice interacting with lists. This is to get them comfortable with list methods and creating code on their own that will be appropriate for the CPT.
- Algorithms: We will go over what an algorithm is and relate algorithms to the real world. We plan on showing a video with a dad asking his kids to make instructions for him to make a PBJ sandwich. We then plan on doing an offline task to have students write instructions to make a paper airplane. What happens when they give these instructions to another group? We will then review some vocab on sequencing, iteration, and selection by pointing to where these exist in a code snippet. This will help tie our students into the next lesson of procedures and help prime them for writing clear and direct instructions.

Placement:

- This unit is placed after a unit on variables and before a unit on procedures.
- The unit on variables went over variable types, declaration and assignment, input and ouput, along with math expressions.
- The unit on procedures goes over procedures, what they are, how to write them, and practice writing them.

Abstractions, Algorithms and Lists

by Victoria Berkowitz and Stephannia Kabanakis

General Overview

(include here description of unit, what class(es) it fits into, when...)

- Abstraction
- Introduction to Lists (What is a list? List methods and indices)
- List Programming
- Algorithms (What is an algorithm and where do we use them in our every day lives?)

Motivation for Unit

(why have you decided to make this?)

- We created this unit because we feel that abstraction pairs well with lists.
- We also added in algorithms as we get more into the idea that computers take specific instructions and being clear and direct is a neccessity.
- After learning strings, students will have many more opportunities to practice programming and interact with the previous lesson of variables as well.
- Including algorithms will help better prepare students with the concepts of procedures and help tie this unit into the next.

Standards Referenced

(select one of the standards sets reviewed in class (CSTA, NY, MA, RI), include a link and a brief explanation as to why you selected that set)

- 9-12.CT.7 Design or remix a program that utilizes a data structure to maintain changes to related pieces of data.
- 9-12.CT.6 Demonstrate how at least two classic algorithms work and analyze the trade-offs related to two or more algorithms for completing the same task.

Tools Used

(include programming language(s), specific programs/environments, and other tools (digital or otherwise) if necessary)

- Hatch Programming Language
- Canvas
- Peardeck

Resources

(include any links/books/readings to be used during this unit)

Lessons

Total length: 2 Weeks

(list each lesson with main topic(s))

- Abstractions
- Lists
- List Programming Practice
- Algorithms
- 1 day of programming for Lists
- Worksheet Guided Lesson
- Review
- End of Unit Quiz
- End of Unit Practice Create Performance Task

Assesments

(list summative and/or formative assessments used)

- Worksheet assignments
- Quiz at the end of the Unit

- Practice Create Performance Task

Assignments

The worksheet assignments have been placed within this folder. The coding assignments are described within this file.

Worksheet Assignments

List Practice Worksheet 1

- This worksheet pairs the APCSP Reference Sheet with list coding problems to solve. It scaffolds and spirals the work to help students better understand and utilize the APCSP Reference sheet.

List Practice Worksheet 2

- This worksheet has no APCSP reference sheet built in, but is instead used as practice to help students get comfortable with lists and manipulating them.

Practice AP Questions

- This assignment is full of AP-styled questions to help students practice and show off their knowledge at the end of the unit.

Coding Assignments

Hatch List Practice

In the Hatch app, practice using list indices by writing a program that completes the tasks below:

Tasks:

To complete each task, you must use the indices from the list.

Create a List: Use the following list in your code,

List Name: Color

List Items:

- red
- orange
- yellow
- green
- blue
- purple

Add to a List:

- Ask the user to enter another color name
- Ask the user to enter an index to insert the color at
- Insert the new color at that index

Remove from a List:

- Ask the user to enter an index number to remove from the list
- Remove that color from the list.

Accessing Value in a list:

- Have the sprite say the first color in the list.
- Have the sprite say the last color in the list.
- Have the sprite day the 4th color in the list.

Hatch Wish List

- This assignment gives students some code that already has the list created for them
- Add 2 items to your list
- This can be anything you would like to buy such as food, games, etc.
- Ask the user what item they would like to add to the wish list. Add this to the list.
- Repeat 3 times
- Tell the user that the first item in your list is out of stock.
- Ask the user what item they would like to replace the first item with.
- Then replace the first item with the item the user input.
- Have your sprite tell the user how long the list it.
- Use a complete sentence.
- Hint: Use the join block and the length of block.

Hatch Number List

- You will only need to change the set blocks here to hold the proper values.
- Follow the sum example
- Find the difference between the 3rd and 4th elements/item of numberList and assign it to the variable named difference
- Find the quotient of the last and 1st elements of numberList and assign it to the variable named quotient
- Find the product of the 2nd and 3rd elements of numberList and assign it to the variable named product
- Perform MOD on the 2nd and 4th element and assign it to the variable named mod

Lesson Plan: Abstractions

Objective:
- Students will define and demonstrate what is an abstraction is and how to use one through coding
- Students will demonstrate how to use variables as abstractions
- Start the Create Performance Task for the AP

Standard:
- 9-12.CT.10 Collaboratively design and develop a program or computational artifact for a specific audience and create documentation outlining implementation features to inform collaborators and users.

Materials:
- Slides

Time Breakdown:
1. Do Now (3 min)
2. Abstraction Lecture/Slides(7 min)
    a. Examples
    b. Variables as Abstractions
    c. Dynamic vs. Static projects
3. Taco Truck Project and how it is an abstraction (5 min)
4. Purpose vs. Functionality (3 min)
    a. Define
    b. Example Vending Machine
5. Start the Create Performance Task using already made Taco Truck Project (Rest of class)

Lesson Plan: Lists

Objective:
- Students will understand what a list is in computer science
- Students will use methods to print, alter and find the length of a list
- Students will create and use a list through code

Standard:
- 9-12.CT.7 Design or remix a program that utilizes a data structure to maintain changes to related pieces of data.

Materials:
- Slides

Time Breakdown:
Student/Group Paced Activities

6. Do Now (3 min)
7. Abstraction Lecture/Slides(7 min)
    a. Examples
    b. Variables as Abstractions
    c. Dynamic vs. Static projects
8. Taco Truck Project and how it is an abstraction
9. Purpose vs. Functionality
    a. Define

    b. Example Vending Machine
10. Start the Create Performance Task using already made Taco Truck Project

Lesson Plan: List Practice Programming Day 1

Objective:
- Students will practice creating and altering lists through coding challenges

Standard:
- 9-12.CT.7 Design or remix a program that utilizes a data structure to maintain changes to related pieces of data.

Materials:
- iPad/Chromebook

Time Breakdown:
Students will work in pairs and the teacher will circulate to answer any questions.
Hatch - List Practice
1. **Create a List:**
    - **Named color with 6 colors in it**
2. **Add to a List:**
    - Ask the user to enter another color name
    - Ask the user to enter an index to insert the color at
    - Insert the new color at that index
3. **Remove from a List:**
    - Ask the user to enter an index number to remove from the list

- Remove that color from the list.

4. **Accessing Value in a list:**
    - Have the sprite say the first color in the list.
    - Have the sprite say the last color in the list.
    - Have the sprite day the 4th color in the list.

<u>Hatch - Wish List</u>

1. Add 2 items to your list
    - i. This can be anything you would like to buy such as food, games, etc.
    - Ask the user what item they would like to add to the wish list. Add this to the list.
        - i. Repeat 3 times
    - Tell the user that the first item in your list is out of stock.
    - Ask the user what item they would like to replace the first item with.
    - Then replace the first item with the item the user input.
    - Have your sprite tell the user how long the list it.
        - i. Use a complete sentence.
        - ii. Hint: Use the join block and the length of block.

Hatch - Number List (Students edit already made code)

1. You will only need to change the set blocks here to hold the proper values.

2. Find the difference between the 3rd and 4th elements/item of numberList and assign it to the variable named difference

3. Find the quotient of the last and 1st elements of numberList and assign it to the variable named quotient

4. Find the product of the 2nd and 3rd elements of numberList and assign it to the variable named product

5. Perform MOD on the 2nd and 4th element and assign it to the variable named mod

Lesson Plan: Algorithms

Objective:
- Students will define and demonstrate what an algorithm is
- Students will create their own algorithms
- Students will understand how algorithms are used in everyday life

Standard:
- Demonstrate how at least two classic algorithms work and analyze the trade-offs related to two or more algorithms for completing the same task.

Materials:
- Slides
- Paper
- Writing Utensils

Time Breakdown:
Student/Group Paced Activities

- Do Now (3 min)
- Introduce Lists (7 min)
    - What methods you can use on them
    - What they look like in code
- Practice AP Questions (5 min)
- List Activity (5 min)
    - Have student construct and alter a list as a class using methods from the slides
- Coding Challenge rest of class to practice

Lesson Plan: Day 2 of programming

Objective:
- Student will be able to:
    - Practice and show their skills in creating list
    - Enhance their knowledge of list while using the different procedures that are encased within Hatch

Standard:
- 9-12.CT.4 Implement a program using a combination of student-defined and third-party functions to organize the computation.
- Design or remix a program that utilizes a data structure to maintain changes to related pieces of data.

Materials:
- Chrome books or ipad
- Mineolahs.oyoclass.com
- Slide deck

Time Breakdown:

- Do Now (3 min)
    - As students come in, there is an entry ticket recapping what different functions/procedures list already have
- Coding (35min)
    - Students are working on completing different list coding assignments
    - They are able to use past slides
    - We use the 3 before me process, where students have to ask 3 other classmates prior to asking me. In addition they need to recap and apply all the suggestions.
- Closure(5)
    - Bring the class together to discuss the challenges they faced and how they were able to overcome it.

Lesson Plan: List Reference Sheet

Objective:
- Students will be able to:
    - Explain the reference sheet
    - Compare and contrast between Hatch & Pseudocode
    - Apply their comprehension through practice AP questions.

Standard:
- 9-12.CT.9 Systematically test and refine programs using a range of test cases, based on anticipating common errors and user behavior.
- 9-12.DL.2 Communicate and work collaboratively with others using digital tools to support individual learning and contribute to the learning of others.

Materials:
- Slides
- List Reference Worksheet
- Writing Utensils

Time Breakdown:
Student/Group Paced Activities
- Introduction/instructions (3minutes)
- Main activity (30min)
    - This activity is a self paced activity where students sign into a peardeck where there are videos created to help explain each section of the AP reference sheet.
    - They then have to answer questions based on that section of the reference sheet
- Closure(5min)
    - Ask the students to reflect on the reference sheet. This allows them to make connections and see where they struggled. It gives a better gage of what we should focus on.

Lesson Plan: Review

Objective:
- Students will be able to:
    - Explain the reference sheet
    - Compare and contrast between Hatch & Pseudocode
    - Apply their comprehension through practice AP questions.

Standard:

- 9-12.CT.9 Systematically test and refine programs using a range of test cases, based on anticipating common errors and user behavior.
- 9-12.DL.2 Communicate and work collaboratively with others using digital tools to support individual learning and contribute to the learning of others.

Materials:
- Slides
- Quizziz
- gimkit
- Review Worksheet

Time Breakdown:
Student/Group Paced Activities
- We are reviewing for the multiple choice portion of the unit wrap up.
- We first review the worksheet, where we start with the ones the students found the most difficult
- We then proceed to move onto the vocabulary quizziz
- Then to gimkit for the concept and more practice multiple choice questions.
-

Lesson Plan: Practice CPT (2 days)

Objective:
- Students will be able to:
    - Develop a program using list and variables while including user input
    - Explain their program and answer questions

Standard:
- 9-12.CT.7 Design or remix a program that utilizes a data structure to maintain changes to related pieces of data.
- 9-12.CT.10 Collaboratively design and develop a program or computational artifact for a specific audience and create documentation outlining implementation features to inform collaborators and users.

Materials:
- Google doc (record their responses)
- PowerPoint (references)
- Chrome book/ ipad (to code)

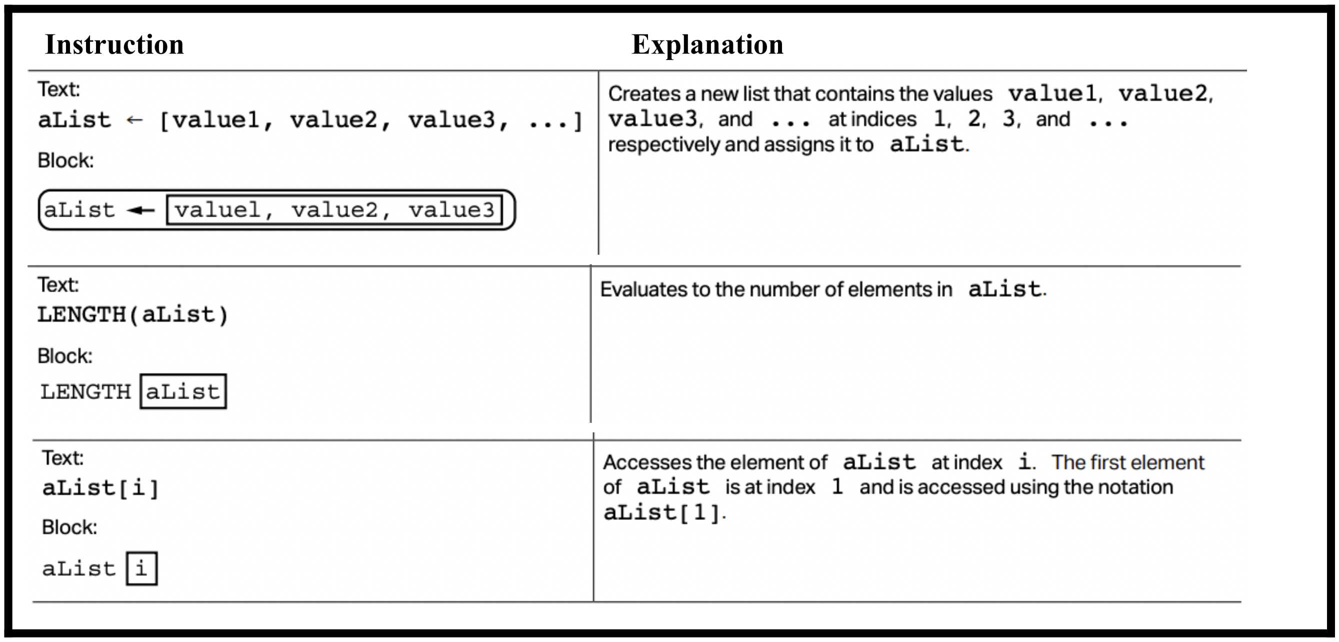
Time Breakdown:
Student/Group Paced Activities
- Introduction/instructions (5 minutes)
    - Go over the requirements for the create task.

- Give them the choice between 4 programs to create:
    1. Random number generator
    2. Fortune teller
    3. Sports roster
    4. Music playlist
- Create Task (37 minutes)
  - Students are to choose a program and make sure the requirements are met.
  - They are able to reference all prior materials and to ask their classmates for help.
  - They use the ask 3 before me rule prior to jumping in.
- Exit Tickets:
  - On the exit ticket we ask:
    1. What was challenging
    2. What was a breeze
    3. What they need more practice with.

Name:_______________ List Worksheet

Creating a List, Assigning Values, Evaluating Length, Accessing Index

Consider the following snippet from the **AP Exam Reference Sheet**:



Instruction	Explanation

Text:
`aList ← [value1, value2, value3, ...]`
Block:
`aList ← value1, value2, value3`
	Creates a new list that contains the values `value1`, `value2`, `value3`, and `...` at indices `1`, `2`, `3`, and `...` respectively and assigns it to `aList`.

Text:
`LENGTH(aList)`
Block:
`LENGTH aList`
	Evaluates to the number of elements in `aList`.

Text:
`aList[i]`
Block:
`aList i`
	Accesses the element of `aList` at index `i`. The first element of `aList` is at index `1` and is accessed using the notation `aList[1]`.

Consider the following code that creates a list called fruits:

```
fruits ← [ "apples" , "bananas" , "oranges" , "peach"]
```

1. Write the result of executing each code segment:

Code	**Output**
`DISPLAY ( fruits [ 2 ] )`	*bananas*
`DISPLAY ( fruits [ 3 ] )`	
`DISPLAY fruits 4`	
`DISPLAY( LENGTH( fruits ) )`	

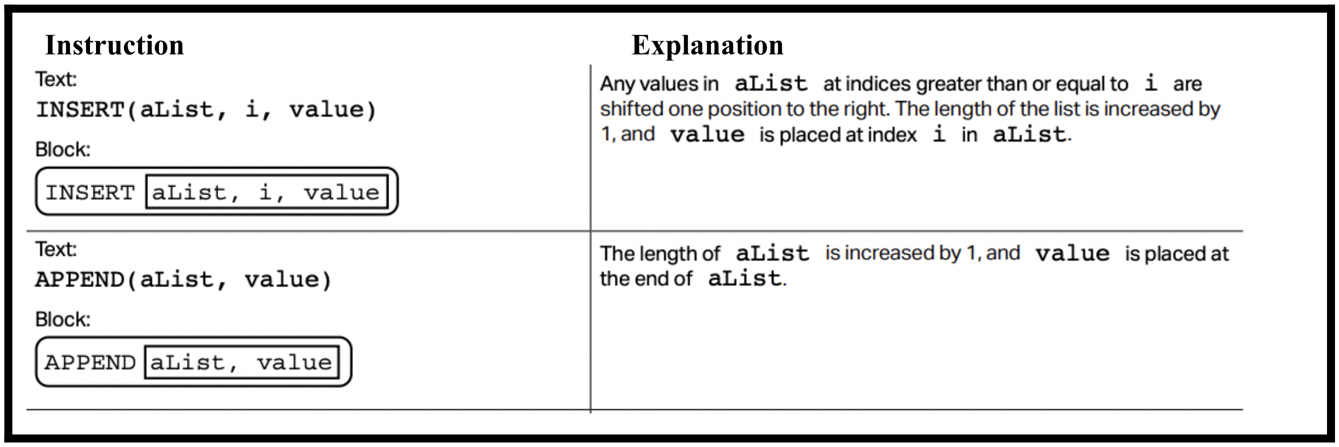
`myChores ← [ ]`
`DISPLAY ( LENGTH ( myChores) )`	

`names ← "Berkowitz", "Knopf", "Kabanakis"`
`DISPLAY LENGTH names`	

Name:______________ List Worksheet

Inserting and Appending Items in a List

Consider the following snippet from the AP Exam Reference Sheet:



Instruction	Explanation

Text:
`INSERT(aList, i, value)`
Block:
`INSERT [aList, i, value]`
	Any values in `aList` at indices greater than or equal to `i` are shifted one position to the right. The length of the list is increased by 1, and `value` is placed at index `i` in `aList`.

Text:
`APPEND(aList, value)`
Block:
`APPEND [aList, value]`
	The length of `aList` is increased by 1, and `value` is placed at the end of `aList`.

2. Look at the chart above, what is the difference between **`INSERT`** and **`APPEND`**?

3. The **INSERT** method takes 3 variables as input. Input variables are called **parameters.** What do each of the parameters represent?

aList ___________________ i ____________________ value ____________________

4. Write out what each list looks like after executing the code segment on the left:

Code	List Contents

`weather ← ["sun", "rain"]`
`APPEND ( weather, "snow" )`
	["sun", "rain", "snow"]

`ages ← [78, 65, 22, 44]`
`INSERT [ages, 2, 14]`	

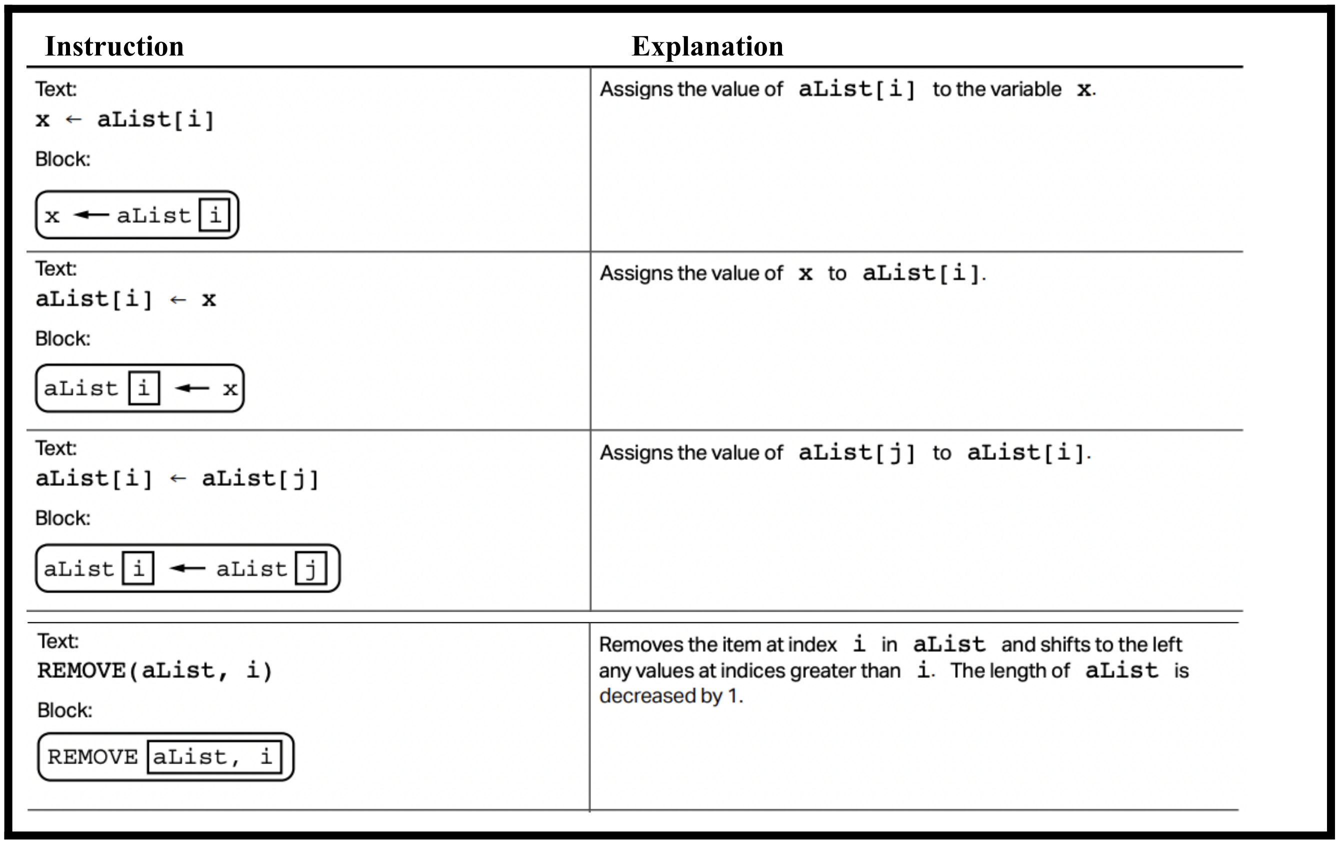
`groceries ← ["eggs" , "cereal" , "rice"]`
`INSERT [groceries, 3 , "milk"]`	

`numbers ← [ 4, 3, 9 ]`
`INSERT ( numbers , 1 , 22)`
`APPEND( numbers , -3 )`	

Name:_______________ List Worksheet

Various Assignments and Removing Items in a List

Consider the following snippet from the AP Exam Reference Sheet:



Instruction	Explanation

Text:
`x ← aList[i]`
Block:
`x ← aList [i]`
	Assigns the value of `aList[i]` to the variable `x`.

Text:
`aList[i] ← x`
Block:
`aList [i] ← x`
	Assigns the value of `x` to `aList[i]`.

Text:
`aList[i] ← aList[j]`
Block:
`aList [i] ← aList [j]`
	Assigns the value of `aList[j]` to `aList[i]`.

Text:
`REMOVE(aList, i)`
Block:
`REMOVE [aList, i]`
	Removes the item at index `i` in `aList` and shifts to the left any values at indices greater than `i`. The length of `aList` is decreased by 1.

5. Write out what each list looks like after executing the code segment on the left:

Code	**List Contents**

`drinks ← ["milk", "soda", "tea"]`
`drinks [ 2 ] ← "water"`
	["milk", "water", "tea"]

`letters ← ["a", "z", "k"]`
`letters [ 1 ] ← "b"`	

`numbers ← [4, 3, 9]`
`REMOVE [numbers , 1]`	

`colors ← ["blue", "pink", "green"]`
`REMOVE (colors , 2 )`
`colors [ 1 ] ← "orange"`
`colors [ 2 ] ← "yellow"`	

PRACTICE QUESTIONS

1. Consider the following code segment:

```
numberList ← [4, 3, 2, 7]
X ← numberList[1] + numberList[3]
DISPLAY(X)
```

What will be displayed after executing the code segment?

A. 6
B. 4
C. 7
D. 9

2. Consider the following code segment:

```
firstList ← ["a", "b", "c"]
secondList ← ["d", "e"]
thirdList ← []
thirdList ← firstList
```

What are the contents of `thirdList` after the code segment is executed?

A. `[]`
B. `["a", "b", "c"]`
C. `["d", "e"]`
D. `["a", "b", "c", "d", "e"]`

3. Consider the following code segment:

```
myList ← [4, 5, 3, 6]
APPEND ( numbersList , 1 )
X ← numbersList[2]
Y ← numbersList[5]
Z ← X + Y
DISPLAY (Z)
```

What displays after the code segment is executed?

A. 7
B. 11
C. 10
D. 6

Name: ____________________ List Worksheet

Code	Output

colors ← ["red", "yellow", "green", "blue"]
DISPLAY(LENGTH (colors))	

colors ← ["red", "yellow", "green", "blue"]
REMOVE (colors, 2)
REMOVE (colors, 3)
DISPLAY(LENGTH (colors))	

colors ← ["red", "yellow", "green", "blue"]
APPEND (colors, "orange")
APPEND (colors, "pink")
DISPLAY(LENGTH (colors))	

colors ← ["red", "yellow", "green", "blue"]
colors [1] ← ["pink"]
colors [4] ← ["white"]
DISPLAY(LENGTH (colors))	

nums ← [5, 3, 9, 2]
DISPLAY(nums [2])	

nums ← [5, 3, 9, 2]
Z ← nums [1] + nums [2]
DISPLAY(Z)	

nums ← [5, 3, 9, 2]
Z ← nums [1] * nums [2]
DISPLAY(Z)	

nums ← [5, 3, 9, 2]
X ← nums [3]
Y ← nums [1]
Z ← Y + X
DISPLAY(Z)	

nums ← [5, 3, 9, 2]
X ← nums [3]
Y ← nums [1]
Z ← Y MOD X
DISPLAY(Z)	

Name: ____________________ List Worksheet

colors ← ["red", "yellow", "green", "blue"]	
What is the index of yellow?	
What is the position of blue?	
What is the index of red?	
What is the index of blue?	

Code	**List after Code Executes**

`animals ← ["cat", "dog", "fish"]`
`APPEND ( animals, "bird" )`
	["cat", "dog", "fish", "bird"]

`animals ← ["cat", "dog", "fish"]`
`APPEND ( animals, "bird" )`
`INSERT ( animals, 2, "horse" )`	

`food ← ["candy", "milk", "fish"]`
`APPEND ( food , "bird" )`
`INSERT ( food , 2, "horse" )`
`REMOVE ( food , 4 )`	

`names ← ["joe", "liam", "alex"]`
`REMOVE ( names, 2 )`	

`ages ← [1, 3, 5]`
`APPEND ( ages, 7 )`
`APPEND ( ages, 9 )`
`ages[ 1 ] ← 0`	

`nums ← [9, 8, 7]`
`nums [ 1 ] ← 10`
`nums [ 2 ] ← 9`
`nums [ 3 ] ← 8`	

`nums ← [9, 8, 7]`
`temp ← nums [ 1 ]`
`nums [ 1 ] ← nums [ 2 ]`
`nums [ 2 ] ← temp`	

Name:_____________ Unit 2:Practice AP Questions

1. Consider the following code segment:

```
myList ← ["red", "yellow", "green", "blue"]
X ← <MISSING CODE>
DISPLAY(X)
```

   What should `<MISSING CODE>` be replaced with so that “yellow” is displayed after executing the code?

   A. `myList[2]`
   B. `myList[“yellow”]`
   C. `“yellow”[myList]`
   D. `[2]myList`

2. Consider the following code segment:

```
numberList ← [2, 5, 8, 1]
X ← numberList[1] + numberList[3]
DISPLAY(X)
```

   What will be displayed after executing the code segment?

   A. 10
   B. 4
   C. 3
   D. 2

3. Consider the following code segment:

```
words ← [“hello”, “bye”, “no”, “yes”]
```

   Which of the following is a valid index for the list, words?

   A. 5
   B. -1
   C. “bye”
   D. 4

4. Consider the following code segment:

```
oceans ← "pacific", "atlantic", "indian"
APPEND oceans, "arctic"
APPEND oceans, "southern"
DISPLAY oceans 4
```

What is displayed after executing the code segment?

A. `"oceans"`
B. `"arctic"`
C. `"southern"`
D. `"indian"`

5. A programmer is writing a sports program that handles football teams and their players. Which of the following is the most appropriate data type to represent a collection of different football player's names?

A. String
B. Boolean
C. Numeric
D. List

6. Consider the following code segment:

```
firstList ← ["apple", "banana", "orange"]
secondList ← ["grape", "lemon"]
thirdList ← []
thirdList ← firstList
firstList ← secondList
secondList ← thirdList
```

What are the contents of `secondList` after the code segment is executed?

A. `[]`
B. `["apple", "banana", "orange"]`
C. `["grape", "lemon"]`
D. `["apple", "banana", "orange", "grape", "lemon"]`

Name:_____________ Unit 2:Practice AP Questions

7. Consider the following code segment:

```
numbersList ← [8, 6, 2, 1]
APPEND ( numbersList , 4 )
X ← numbersList[1]
Y ← numbersList[5]
Z ← X MOD Y
DISPLAY (Z)
```

What displays after the code segment is executed?

A. 4
B. 0
C. 3
D. 1

8. Which of the following is a benefit of using a list as a data abstraction in a program?

A. Lists often allow their size to be easily updated to hold as many data values as needed.
B. Lists convert all elements to strings so that they can be inspected character-by-character.
C. Lists prevent duplicate data values from appearing in the list.
D. Lists are used to store all input data so that there is a running record of all user input.

9. A programmer is creating a program that will add two user inputs together and display the sum to the user. Which of the following would be an appropriate algorithm?

A. Ask the user for inputs. Display the output.
B. Ask the user for inputs. Store the inputs into variables. Calculate the sum of the two variables. Display the sum.
C. Add the numbers together. Display the numbers.
D. Add numbers together. Store user input into variables. Display the numbers.

10. A student is creating a program that will calculate the average for her friends' grades. Which of the following would be an appropriate algorithm?

A. Create a list for grades, ask the user for input, store the user input into the list, calculate the average of the list, return to the user
B. Ask the user for input, store the user input into the list, calculate the average of the list, create the list for grades
C. Create a list for grades, return the output to the user, ask the user for input, calculate the average of the list,
D. Ask the user for input, return to the user, create the list for grades, calculate the average for the list