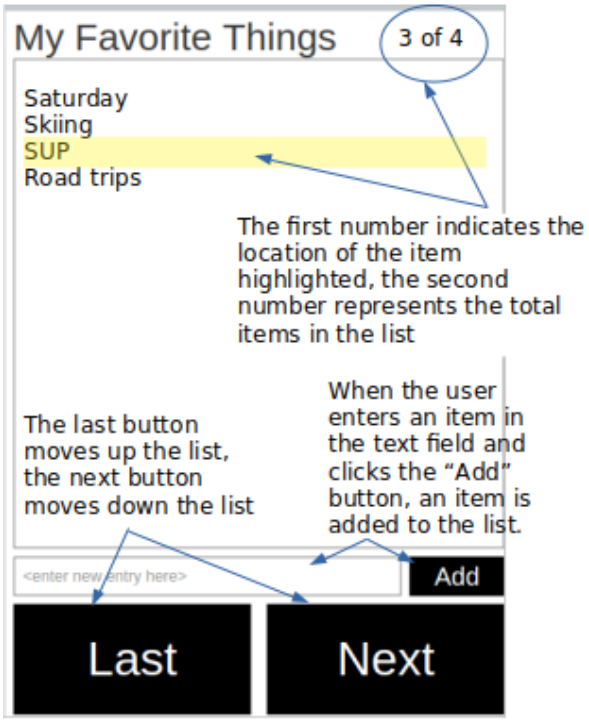
Name _____Period _____

My Favorite Things

Your Tasks (Mark these off as you go)

- ☐ Review the My Favorite Things App
- ☐ Write the addItem function
- ☐ Write the displayItems function
- ☐ Write the scroll function
- ☐ Receive credit for this lab guide

☐ Review the My Favorite Things App



The image shown to the right represents a Graphical User Interface (GUI) for which you must make work as intended. A description of how each feature should function is provided.

The code for each feature should be organized into its own function. For example, the text which indicates "3 of 4" should be updated in its own function. There should also be function that controls the highlighting of the list items when last and next are clicked.

.

☐ Write the addItem function

A user can add items to the list by typing the item they want to add in the text box and clicking the add button. Recall that to add an item to an array, you use the *push* command. Below are a few examples,

Code	Output

`var groceryList = [] ;`
 	
`groceryList.push("apples");`
`groceryList.push("carrots");`
 	
`console.log(groceryList);`
	▼ *(2) ["apples", "carrots"]*
	0: "apples"
	1: "carrots"
	length: 2

```
var primeNumbers = [];

primeNumbers.push(3);
primeNumbers.push(5);

console.log(primeNumbers);
```

```
▼ (2) [3, 5]
    0: 3
    1: 5
    length: 2
```

Also, recall that we can get a value from an input field as follows,

```
var getInput = document.createElement("input");
document.body.append(getInput);
```

← Creates an input field

```
var showButton = document.createElement("button");
showButton.style.width = 50 + "px";
showButton.innerHTML = "Show";
document.body.append(showButton);
showButton.addEventListener("click", showContents);
```

← Creates a button

```
function showContents(){
console.log(getInput.value);
}
```

← Displays the contents of the input field

The following code creates the input field for your app,

```
var listInput = document.createElement("input");
listInput.placeholder = "add an item";
listInput.style.position = "relative";
listInput.style.cssFloat = "left";
listInput.style.top = "32px";
listInput.style.width = "80%";
container.append(listInput);
```

Consider the array declared below,

```
var favsList = [];
```

(a) Each time add button is clicked, the *addItem* function is called. In the *addItem* function, write code that could be used to add an item to the *favsList* array.

(b) Each time an item is added, it needs to display on the screen. You will write this function in a moment, but for right now, below the code your wrote for part (a) call *displayItems()*.

(c) Finally, we will need to keep track of the index of each item we add. The count variable declared below will help us keep track of that. At the bottom of the addItem() increment count by 1.

```
var count = 0;
```

```
function addItem(){




}
```

□ Write the displayItems function

We can access items in an array, by indicating the name of the array, followed by the index of the location we want to retrieve in brackets. Notice in the below example, we begin our indexing at 0. That is apples is at location 0, carrots is at location 1, and coffee is at location 2.

Code	Output

`var groceryList = [] ;`
`groceryList.push("apples");`
`groceryList.push("carrots");`
`groceryList.push("coffee");`
 	
`console.log(groceryList[0]);`
`console.log(groceryList[1]);`
`console.log(groveryList[2]);`
	apples
	carrots
	coffee

The code below creates the container that will display the items the user adds,

```
var listContainer = document.createElement("div");
listContainer.style.border = "solid";
listContainer.style.borderColor = "grey";
listContainer.style.backgroundColor = "LightGrey";
listContainer.style.position = "relative";
listContainer.style.height = "80%";
listContainer.style.top = "30px";
listContainer.style.textAlign = "left";
container.append(listContainer);
```

In the body of the display items function, do the following,

(a) Create a new paragraph element
(b) Assign the id of the element to count. This will enable us to tell the different items apart on the screen
(c) Set the innerHTML of the paragraph element to the favsList item at position count.
(d) Append the paragraph element you created to the listContainer
(e) Reset the listInput value as shown below. This will cause the text in the input field to go away after each item is added. That way the user doesn't need to delete it to add another item.

```
listInput.value = "";
```

□ Write the scroll function

The next and last buttons enable the user to scroll through the items on the list. If the next button is clicked, the next item is highlighted. If the last item is clicked, the previous button is highlighted.

The code below creates the next and previous buttons,

```
//Creates the previous button.  When the previous button
//is clicked the previous item in the list will
//be highlighted on the screen
var previousButton = document.createElement("button");
previousButton.id = "previous";
previousButton.innerHTML = "previous";
previousButton.style.position = "relative";
previousButton.style.cssFloat = "left";
previousButton.style.top = "32px";
previousButton.style.width = "50%";
container.append(previousButton);

//Creates the next button.  When the next button
//is clicked the next item in the list will
//be highlighted on the screen
var nextButton = document.createElement("button");
nextButton.id = "next";
nextButton.innerHTML = "next";
nextButton.style.position = "relative";
nextButton.style.cssFloat = "right";
nextButton.style.top = "32px";
nextButton.style.width = "50%";
container.append(nextButton);
```

When either one of the buttons are clicked the scroll function is called.

```
previousButton.addEventListener("click", scroll);
nextButton.addEventListener("click", scroll);
```

The currentIndex variable will help us keep track of where we are in our list,

```
var currentIndex = -1;
```

For the scroll function to work properly, do the following,

(a) We first need to get the id of the button that was clicked. Recall we can do this using the code below. If the id the button clicked is equal to "next" the currentIndex needs to increment by 1.

```
var idOfButtonClicked = event.target.id
```

(b) If the id the button clicked is equal to "previous" the currentIndex needs to decrement by 1

(c) The currentIndex corresponds to the items displayed on the page. To style the background of the element, you will need to get the element by that id

```
document.getElementById(currentIndex)
```

and style the backgroundColor to yellow

(d) Finally we will need to update the innerHTML of the locationCount text which displays in the top right corner of the GUI. Currently it says 0 or 0, but should say 1 of favsList.length, 2 of favsList.length, etc.

```
var locationCount = document.createElement("h2");
locationCount.innerHTML = "0 of 0";
```

```
function scroll(event){




}
```

□ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.