

OLI Assignment Due by 9:00pm on Sunday 11/7 via Cougar Courses

As you read the following OLI pages and complete the interactive activities, capture the screenshots of the completed activities and replace the respective screenshots in the document.

- Page 44 Array updates
- Page 45 Array as parameters

When you are ready to submit the assignment, download the document in PDF and submit the PDF file on Cougar Course as the proof for your work.

Page 44 Array updates

LBD remove "at"

What is the value `i` after the initialization portion of the loop?

Since $2 < 4$ is true, which shifting will be performed in the loop body? `words[2] = words[3];` ▼

✓ Correct! Yes, the pos has the value 2.

✓ Correct, since `i` is 2, `i + 1` is 3.

After the above shifting, the compiler will increase the value of `i` from 2 to 3. Since $3 < 4$ is true, the loop body will be executed. What would be shifted this time?

- ☒ `words[3] = words[4];`
- ☐ `words[4] = words[5]`
- ☐ `words[2] = words[3];`

✓ Correct, the left side of `=` uses the value of `i` as its index while the right side of `=` uses the value of `i + 1` as its index.

Will the loop perform more shifting? Why?

Not anymore since now $4 < 4$ is false

Resubmit

✓ After the above shifting, the compiler will increase `i` from 3 to 4. Since $4 < 4$ is false, the loop is now completed.

Now let's test the loop that iterates through the indices on the **right** side of the assignments:

```
for (int i = pos + 1; i < num_words; index++) {  
    words[i - 1] = words[i];  
}  
num_words --;
```

What is the value `i` after the initialization portion of the loop?

Since $3 < 5$ is true, which shifting will be performed in the loop body?? `words[2] = words[3];`

✓ Correct! The value of pos is 2, therefore, pos + 1 is 3. This should be the index of the first element that needs to be moved.

✓ Correct, the index on the left is i - 1 and the index on the right is i.

After the above shifting, the compiler will increase the value of i from 3 to 4. Since $4 < 5$ is true, the loop body will be executed. What would be shifted this time?

- ☐ words[4] = words[5]
- ☒ words[3] = words[4];
- ☐ words[2] = words[3];

✓ Correct, the left side of = now uses the value of i - 1 as its index while the right side of = uses the value of i as its index.

In your own words, explain why the loop condition is set to `i < num_words` for this approach.

i indicates the index that's being increased by one.

Resubmit

✓ The value of i is used to control the index of the element to be shifted. The last element to be shifted is words[num_words - 1]. Setting `i < num_words` will make sure that we don't attempt to shift words[num_words].

Hotspot find where str is in the words array

Before the loop, the code segment sets `pos` to the value of `num_words`, which is 5. In addition, the value of `found` is set to `false` before the loop.

Suppose "a" is the value of `str`.

How many times will the comparison `(str == words[i])` be evaluated by the loop?

When the value of `found` is set to true, what is the value of `i`?

After the loop, what is the value of `pos`?

✓ Correct. since "a" is the same as `words[0]`, the conditional statement will set `found` to true after evaluating `(str == words[0])`. This would cause `!found` to be false and break out of the loop.

✓ Correct. `found` is set to true when `str == words[0]` evaluates to true.

✓ Correct. the variable `pos` is set to 0 when `str == words[0]` evaluates to true.

Suppose "at" is the value of `str`.

How many times will the comparison `(str == words[i])` be evaluated by the loop?

When the value of `found` is set to true, what is the value of `i`?

After the loop, what is the value of `pos`?

✓ Correct. the loop will evaluate `str == words[0]`, `str == words[1]`, `str == words[2]`. The first two evaluations result in false. The last one results in true and will set `found` to true. This would cause `!found` to be false and break out of the loop.

✓ Correct. `found` is set to true when `str == words[2]` evaluates to true.

✓ Correct. the variable `pos` is set to 2 when `str == words[2]` evaluates to true.

Suppose "bat" is the value of `str`.

How many times will the comparison `(str == words[i])` be evaluated by the loop?

After the loop, what would be the value of `found` after the loop?

After the loop, what would be the value of `pos`?

✓ Correct. the loop will compare `str` to all five elements because none of them is a match for "bat".

✓ Correct. since no match is found, the if-block of the conditional statement in the loop body was never executed. Therefore, the value of `pos` remains unchanged from before the loop.

✓ Correct. since no match is found, the if-block of the conditional statement in the loop body was never executed. Therefore, the value of `pos` remains unchanged from before the loop.

Click [this link](#) to visit a program that has implemented the removal of element by value. Run the program with different words of your choice.

What questions or tips for others do you have regarding the removal of elements from an array.

None.

Resubmit

✓

LBD shifting elements

The above video discussed the importance of shifting from the back of the array. How should we order the following statements in order to properly shift the elements before adding "ace" to words[1]?

<input type="checkbox"/>	words[5] = words[4];
<input type="checkbox"/>	words[4] = words[3];
<input type="checkbox"/>	words[3] = words[2];
<input type="checkbox"/>	words[2] = words[1];

Check My Answer

✓ Correct, we need to start shifting from the end; "tac" first, "cat" second, "at" third, and "act" last.

Hotspot two alternatives in shifting

First, answer the following two questions for pos being 3.

What is the first shifting that would occur?

- ☒ `words[5] = words[4];`
- ☐ `words[4] = words[3];`
- ☐ `words[4] = words[5];`

✓ Correct. The first shifting is always moving the last element.

What is the last shifting that would occur?

- ☐ `words[3] = words[2];`
- ☐ `words[3] = words[4];`
- ☒ `words[4] = words[3];`

✓ Correct. The last shifting is always to make room for the new element.

Now, answer the following two questions for pos being 0.

What is the first shifting that would occur?

- ☒ `words[5] = words[4];`
- ☐ `words[1] = words[0];`
- ☐ `words[2] = words[1];`

✓ Correct. The first shifting is always moving the last element.

What is the last shifting that would occur?

- ☒ `words[1] = words[0];`
- ☐ `words[0] = words[1];`
- ☐ `words[2] = words[1];`

✓ Correct. The last shifting is always to make room for the new element.

Hotspot insert an element by value

Before the loop, pos is set to 0 and found is set to false. The loop initialization assigns the value of num_words to i, which is 5. Therefore, $i > 0 \ \&\& \ !\text{found}$ evaluates to true and the loop body will be executed.

What is the result of `words[i - 1] > str`?,

Will there be any shifting of the element?

What would be the value of pos after the loop?

✓ Correct, since i is 5, words[i - 1] is "tac", which is smaller than "zip".

✓ Correct, since words[i - 1] > str is false, no shifting is performed.

✓ Correct, since words[i - 1] > str if false, the else-block will be executed and set pos to the value of i.

Now let's first trace the above code segment using the str value of "ace".

Before the loop, pos is set to 0 and found is set to false. The loop initialization assigns the value of num_words to i, which is 5. Therefore, $i > 0 \ \&\& \ !\text{found}$ evaluates to true and the loop body will be executed.

What is the result of `words[i - 1] > str`?,

What shifting would be performed?

What is the value of i when words[i - 1] > str evaluates to false?

What would be the value of pos after the loop?

✓ Correct, i has the value 5, words[i - 1] is "tac", "tac" is larger than "ace".

✓ Correct, the index on the left is i and the index on the right is i - 1.

✓ Correct, since words[0] > str is false, the else-block will be executed and set pos to the value of 0 + 1.

✓ Incorrect, when i is 0, the loop condition is false and the conditional statement will not be executed.

Click on [this link](#) to access a program that includes the above code. Run the program with your choices of new words. Feel free to adjust the initialization list for the words array.

What questions or tips for others do you have regarding the insertion of elements into an array?

None.

Resubmit

✓ Thanks for sharing.

MR access invalid element

Share an experience when you realized that you had accessed an invalid element of an array. Provide as much details as you could to help us understand the situation.

Haven't had one yet.

Resubmit

✓ Thanks for sharing.

If you don't think you have ever done that, tell us what you think could happen when a program is using an index that is outside of the proper range for the given array.


Errors and infinite. If not that then you'll get the wrong output you desire.

Resubmit


✓ Thanks for sharing.

Page 45 Arrays as parameters

Hotspot prototypes of functions that work with arrays

When specifying that a parameter is an array, we put [] after  the parameter name.

✓ Correct.

When specifying an array parameter in the function prototype, we put nothing  in [].

✓ Correct.

Mark all functions that will **not** update the elements of their array parameter.

☐ collect

☒ count

☐ triple

☒ find

☒ print

Check My Answer

✓ Correct, these functions add the keyword `const` to guarantee no change be made to the elements of `arr`.

Hotspot calling functions with an array parameter

Given the above function prototype, complete the following function call to display the first three elements of the `highs` array.

`print(` `, 0, 3);`

✓ Correct, we just need to tell the function which array to process.

```
void print(const int arr[], int n);
```

Given the above function prototype, complete the following function call to display the first three elements of the `highs` array.

`print(highs,` `);`

✓ Correct! This argument specifies the number of elements to be displayed.

```
void print(const int arr[], int start, int end);
```

Given the above prototype, complete the following function call to display the **last three elements** of the `lows` array. As a reminder the first element of the `lows` array is `lows[0]` while the last element of the `lows` array is `lows[6]`.

`print(lows ,` `,` `);`

✓ Correct! This argument specifies the index for the first element to be displayed by the function. The last three elements of `highs` are `lows[4]`, `lows[5]`, `lows[6]`.

✓ Correct! This argument specifies the index for the last element to be displayed by the function. The last three elements of `highs` are `highs[4]`, `highs[5]`, `highs[6]`.

How many elements would be displayed by the following function calls?

`print(highs, 4):`

`print(highs, 1, 3):`

✓ Correct! The second argument specifies the number of elements to be displayed.

✓ Correct! The function will display `highs[1]`, `highs[2]`, and `highs[3]`.

LBD display a range of elements

Using the same approach we can implement a few other array-based functions.

```
int calculate_total(const int arr[], int n) {  
    int total = 0;  
    for (int i = 0; i < n; i++) {  
        //to update total  
    }  
    return total;  
}
```

The above function is to return the sum of all values stored in the arr array. In the following box, write the statement that could be used in the above loop body to update total.

```
total += arr[i];
```

Resubmit

✓ total += arr[i];

```
bool find(const int arr[], int start, int end; int value) {  
    bool found = false;  
    for (int i = start; i <= end & !found; i++) {  
        //to compare an element with value and update found  
    }  
    return found;  
}
```

The above function is to return true if the given value is between arr[start] and arr[end], false otherwise. In the following box, write the statement that could be used in the above loop body.

```
if (arr[i] == value) {  
    found = true;  
}
```

Resubmit

✓

```
if (arr[i] == value) {  
    found = true;  
}
```

Hotspot function to remove an element by position

Which of the following is the right call to remove the first element from the `highs` array?

- ☒ `remove(highs, num_highs, 0);`
- ☐ `remove(highs, CAPACITY, 0);`
- ☐ `remove(highs, 7, 0);`

✓ Correct, this would remove 69 from the array and reduce `num_highs` by 1.

What would be the value of `num_highs` after the above remove function call?

6

✓ Correct! The remove function reduces the value of the argument for `n` by 1.

Which of the following should be the function prototype for removing an element by value?

- ☐ `void remove(int arr[], int n, int value);`
- ☒ `void remove(int arr[], int& n, int value);`
- ☐ `void remove(const int arr[], int& n, int value);`
- ☐ `void remove(int arr[], int& n, int& value);`

✓ Correct; this would allow `arr` to be updated as well as allowing the change to `n` be reflected upon the completion of the function.

In the following space, define the prototype of a function to insert one element into an array of your choice.

```
void insert(int arr[], int& n, int pos, int value);  
void insert(string arr[], int& n, int pos, string value);  
void insert(double arr[], int& n, double value);
```

Resubmit

void insert(int arr[], int& n, int pos, int value);
✓ void insert(string arr[], int& n, int pos, string value);
void insert(double arr[], int& n, double value); // assume sorted array