

Name(s) _____ **Answer Key** _____ Period _____ Date _____

Activity Guide - Writing Algorithms with Arrays



Algorithms and the Problem-Solving Process

The **Problem Solving Process** is useful when planning and writing algorithms. This process will help you clarify and break down a problem into manageable steps so you can easily identify the code you need to write for each step.

Define

- Read the instructions carefully to make sure you understand the goals.
- Rephrase the problem in your own words.
- Identify any new skills you are being asked to apply.
- If there is starter code, read it to understand what it does.

Prepare

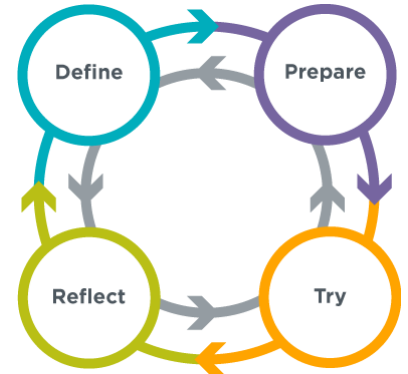
- Write out or draw the steps you need to take to solve the problem.
- List what you already know how to do and what you don't yet.
- Explain your algorithm to a classmate.
- Review similar programs that you've written in the past.

Try

- Write one small piece at a time.
- Test your program often.
- Use comments to document what your code does.
- Go back to a previous step if you get stuck or don't know whether you've solved the problem.

Reflect

- Compare your program to the defined problem to make sure you've solved all aspects of the problem.
- Ask a classmate to try your program and note places where they struggle or show confusion.
- Ask a classmate to read your code to make sure that your documentation is clear and accurate.
- Try to "break" your program to find types of interaction or input that you could handle better.
- Identify changes or improvements you can make next to your program.



Algorithm Planning

For your chosen problem, apply the Problem Solving Process to make sure you understand its requirements, identify similar problems, and plan and refine your algorithm.

Define

Read the instructions carefully to ensure you understand the goals. Rephrase the problem in your own words and identify the **precondition(s)** and **postcondition(s)** of the problem.

The goal of the program is to calculate the total number of AP Computer Science exams taken in each state by adding the segments of the array `csExams` together.

First, call the `FileReader` to read the file and store the result into a `ID` array. Second, instantiate an object for the `APexam` array. Third, call the method that adds them all together to get the result.

Precondition - the array is a list of ints.

Postcondition - the result must be an int and it is the summation of all the integers in the list.

Prepare

Write out the steps you need to take to solve the problem.

Questions to Consider:

- Which steps do you already know how to do? Which don't you know how to do yet?
- Explain your steps to a classmate. Was there anything your classmate found confusing? Did they have suggestions that could help improve your algorithm?
- Look at similar programs that you've written in the past. What did you do before that would be helpful in solving this problem?

Steps I know how to do - write a method for calculating the total number of exams by using a for loop to add each value of the next one and compile the exams together.

I may need help calling the file reader method and explaining how it works in the program.

I talked with a classmate and they were able to understand the steps I shared with them.

When I look at similar programs from past assignments, I will try to find one that calculates the total number of something so that it can help me start my code.

Try

Write one small piece at a time.

Test your program often!

Use comments to document what your code does.

Go back to the previous steps if you get stuck or aren't sure whether you've solved the problem.

Reflect

Compare your finished program to the defined problem to make sure you've solved all aspects of the problem.

Questions to Consider:

- Ask a classmate to try your program. Are there any parts of your code where they were confused?
- Ask a classmate to read your code to make sure that your documentation is clear and accurate. Do they understand what your code is supposed to do?
- What are some changes or improvements you could make to your program?

What have you accomplished?

I broke down my code to find the preconditions and postconditions to help me use arrays and loops in my program.

I wrote code to calculate the array totals by adding the items together, which then helped me to calculate the average of the exams.

What do you need to do next?

I need to check the loops and arrays in my program to make sure that they are used correctly.

Choice A: AP Computer Science Exams

The AP Computer Science Exams dataset includes the number of exams taken in each state in 2021. The number of exams taken in each state is stored in an `int` array called `csExams`.

Write the `calcTotalExams()` method to calculate and return the **total** number of AP Computer Science exams taken in all states.

List of ints that contain the exams for the AP computer Science exams.

Precondition(s)

List of strings that contain the state names.

Postcondition(s)

The calculated number of total exams taken in each state as in `int`.

Pseudo Code

```
totalExams = 0
loop from 0 to end of csExams
  totalExams += csExams[index]
return totalExams
```

```
-----

/*
 * Returns the total number of AP CS exams taken in all states
 */
public int calcTotalExams() {
  /* ----- TO DO -----
   *  Traverse csExams to calculate and return the total number of AP CS exams
   * taken in all states.
   * -----
  */
  int totalExams = 0;

  for (int index = 0; index < csExams.length; index++) {
    totalExams += csExams[index];
  }

  return totalExams;
}
```

Write the `calcAverageExams()` method to calculate and return the **average** number of AP Computer Science exams taken in each state.

Precondition(s)

The int value for the total number of AP computer Science exams.
The list of int values that has a certain length.

Postcondition(s)

The average score for each exam as an int.

Pseudo Code

```
totalExams = calcTotalExams()
averageExams = totalExams / csExams.length

-----

/*
 * Returns the average number of AP CS exams taken in each state
 */
public int calcAverageExams() {
    /* ----- TO DO -----
     *  Calculate and return the average number of AP CS exams taken in each state.
     * -----
    */

    int totalExams = calcTotalExams();
    int averageExams = totalExams / csExams.length;

    return averageExams;
}

/*
 * Returns a String containing each state name and the number of AP CS exams taken
 */
public String toString() {
    String result = "";

    for (int index = 0; index < states.length; index++) {
        result += states[index] + ": " + csExams[index] + " AP CS exams taken\n";
    }

    return result;
}
```

Choice B: Song Plays

The Song Plays dataset includes the number of times a song was played on a streaming music service each day in a month. The number of times a song was played is stored in an `int` array called `songPlays`.

Write the `calcTotalPlays()` method to calculate and return the **total** number times the song was played on the streaming music service in the month.

Precondition(s)

The int array for the number of times a song was played..

Postcondition(s)

The calculated total for the number of times a song was played as an int.

Pseudo Code

```
totalPlays = 0
loop from 0 to end of songPlays
    totalPlays += songPlays[index]
return totalPlays
```

```
-----

/*
 * Returns the total number of times the song was played
 */
public int calcTotalPlays() {
    /* ----- TO DO -----
     *  Traverse songPlays to calculate and return the total number of times the
     * song was played on the streaming music service.
     * -----
    */

    int totalPlays = 0;

    for (int index = 0; index < songPlays.length; index++) {
        totalPlays += songPlays[index];
    }

    return totalPlays;
}
```

Write the `calcAveragePlays()` method to calculate and return the **average** number of times the song was played on the streaming music service each day of the month.

Precondition(s)

The int calculated total for the total number of times a song was played on a streaming music service each day of the month.
The list of int values that has a certain length.

Postcondition(s)

The calculated average for the number of times a song was played on a streaming music service each day in a month.

Pseudo Code

```
totalPlays = calcTotalPlays()
averagePlays = totalPlays / songPlays.length

-----
/*
 * Returns the average number of times the song was played each day
 */
public int calcAveragePlays() {
    /* ----- TO DO -----
     *  Calculate and return the average number of times the song was played.
     * -----
    */

    return -1;
}

/*
 * Returns a String containing the number of plays the song had each day
 */
public String toString() {
    String result = title + " Daily Plays\n-----\n";

    for (int index = 0; index < songPlays.length; index++) {
        result += "Day " + (index + 1) + ": " + songPlays[index] + "\n";
    }

    return result;
}
```

Choice C: Profits

Calculate the sum of the positive profits made by a store.

The Store Profits dataset includes the profits a store made each day of a month. The profits made each day are stored in a double array called `storeProfits`.

Write the `calcTotalProfits()` method to calculate and return the **total** profits made by the store in the month.

Precondition(s)

The int array for the number of positive profits a store made each day of the month.

Postcondition(s)

The calculated total for the number of profits a store made each day for a month and as an int.

Pseudo Code

```
totalProfits = 0
loop from 0 to end of storeProfits
  totalProfits += storeProfits[index]
return totalProfits
```

```
-----

/*
 * Returns the total profits made by the store
 */
public double calcTotalProfits() {
  /* ----- TO DO -----
   *  Traverse storeProfits to calculate and return the total profits made.
   * -----
  */

  double totalProfits = 0;

  for (int index = 0; index < storeProfits.length; index++) {
    totalProfits += storeProfits[index];
  }

  return totalProfits;
}
```

Write the `calcAverageProfits()` method to calculate and return the **average** profits made by the store each day in the month.

The int calculated total for the average profits made by the store each day in the month.

Precondition(s)

The list of int values that has a certain length.

Postcondition(s)

The calculated average profits made by the store each day in the month and returns the value.

Pseudo Code

```
totalProfits = calcTotalProfits()
```

```
average = totalProfits / storeProfits.length
```

```
-----  
  
/*  
 * Returns the average profits made by the store each day  
 */  
  
public double calcAverageProfits() {  
    /* ----- TO DO -----  
    *  Calculate and return the average profits made by the store each day.  
    * -----  
    */  
  
    double totalProfits = calcTotalProfits();  
    double averageProfits = totalProfits / storeProfits.length;  
    return averageProfits;  
}  
  
/*  
 * Returns a String containing the profits the store made each day  
 */  
  
public String toString() {  
    String result = name + "'s Daily Profits\n-----\n";  
    for (int index = 0; index < storeProfits.length; index++) {  
        result += "Day " + (index + 1) + ": " + storeProfits[index] + "\n";  
    }  
    return result;  
}  
}
```

Choice D: TV Episodes Length

The TV Episodes dataset includes the number of minutes for each TV episode in a season. The number of minutes for each episode is stored in an `int` array called `episodeLengths`.

Write the `calcTotalLength()` method to calculate and return the **total** number of minutes for all TV episodes in a season.

Precondition(s) The int array for the number of minutes for each TV episode in a season.

Postcondition(s) Takes the total number of for each TV episode in a season and returns the value
The calculated total for the number of for each TV episode in a season as an int.

Pseudo Code

```
totalMins = 0
loop from 0 to end of episodeLengths
    totalMins += episodeLengths[index]
return totalMins
```

```
-----

/*
 * Returns the total number of minutes for all episodes
 */
public int calcTotalLength() {
    /* ----- TO DO -----
     *  Traverse episodeLengths to calculate and return the total number of
     * minutes for all TV episodes.
     * -----
    */

    int totalLength = 0;

    for (int index = 0; index < episodeLengths.length; index++) {
        totalLength += episodeLengths[index];
    }

    return totalLength;
}
```

Write the `calcAverageLength()` method to calculate and return the **average** number of minutes of each TV episode in a season.

Precondition(s) The int calculated total for the average number of minutes of each TV episode in a season.
The list of int values that has a certain length.

Postcondition(s) The calculated average number of minutes of each TV episode in a season.

Pseudo Code

```
totalMins = calcTotalLength()
average = totalMins / episodeLengths.length
```

```
-----
/*
 * Returns the average number of minutes for each episode
 */
public int calcAverageLength() {
    /* ----- TO DO -----
     *  Calculate and return the average number of minutes for each TV episode.
     * -----
     */

    int totalLength = calcTotalLength();
    int averageLength = totalLength / episodeLengths.length;

    return averageLength;
}

/*
 * Returns a String containing the name and length of each episode
 */
public String toString() {
    String result = "";

    for (int index = 0; index < episodeTitles.length; index++) {
        result += episodeTitles[index] + ": " + episodeLengths[index] + " minutes\n";
    }

    return result;
}
```