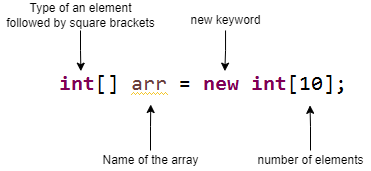
*CSE 102*

**ARRAYS**

A Quick Recap:

* An array is a sequence of fixed-length data items each having the same type and stored contiguously in memory.
* Once created, the size of the array cannot be changed.
* The name (identifier) of an array is a reference to its first element.
* We use arrays to store data which will be necessary in the next stages of our computation.
* Note that arrays are located in main memory, which typically has around 8-16 GB of storage capacity nowadays. Your program can allocate only a part of this storage as there are always other programs running on your computer.
* In order to keep memory usage low, a program should deallocate an array if it won’t be used in future before allocating new ones. This task is done by the programmer in languages like C, C++; but languages like Java has what is called “garbage collector” facility. So you don’t need to worry about deallocation while you’re writing code in Java.

Java Syntax for Arrays:



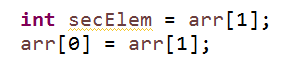
The left hand-side of the assignment operator declares a new variable “arr” which is a reference to an integer array.

The right hand-side actually creates a new array of size 10, by allocating the required amount of space from memory and then initializes its elements to the default value, 0 for int type.

If you want to initialize your array with different values, you can use the following syntax:



Accessing and altering:



**Exercises**:

1. What will be the output of the following program?

**public** **static** **void** main(String[] args){

**int**[] arr = **new** **int**[10];

System.***out***.println(arr[5]);

}

1. undefined
2. 10
3. 0
4. 5
5. What will be the output of the following program?

**public** **static** **void** main(String[] args){

**int**[] arr = **new** **int**[10];

System.***out***.println(arr[10]);

}

1. 0
2. Runtime Error
3. 10
4. Compiler Error
5. What will be the output of the following program?

**public** **static** **void** main(String[] args){

**int**[] arr = **new** **int**[10];

arr[4] = 5;

**int**[] arr2 = arr;

arr2[4] = 3;

System.***out***.println(arr[4]);

}

1. 3
2. 5
3. 10
4. Compiler Error
5. What will be the output of the following program?

**static** **void** cleaner(**int**[] arr) {

**for**(**int** i=0; i<arr.length; i++)

arr[i] = 0;

}

**public** **static** **void** main(String[] args){

**int**[] arr = **new** **int**[10];

arr[0] = 6;

*cleaner*(arr);

System.***out***.println(arr[0]);

}

1. 6
2. 0
3. 10
4. Compiler Error
5. The following function is supposed to print the elements of an integer array to standard output. What should be replaced with question marks?

**static** **void** printArr(**int**[] arr) {

**for**(**int** i=0; i< ??? ; i++)

System.***out***.print(arr[i]+" ");

}

1. len(arr)
2. arr.length()
3. arr.length(i)
4. arr.length
5. The following function is supposed to reverse the elements of an integer array. What should be replaced with question marks?

**static** **void** reverse(**int**[] arr) {

**int** temp;

**int** len = arr.length;

**for**(**int** i=0; i< ??? ; i++) {

temp = arr[i];

arr[i] = arr[len - 1 - i];

arr[len - 1 - i] = temp;

}

}

1. len/2
2. len
3. len/3
4. 2\*len
5. Inspect the following alternative reverse function and compare it to the one given in previous question. Which of the following is false about the comparison?

**static** **int**[] reverseAlt(**int**[] arr) {

**int**[] newArr = **new** **int**[arr.length];

**for**(**int** i=0; i<arr.length; i++) {

newArr[i] = arr[arr.length - 1 - i];

}

**return** newArr;

}

1. reverse() does not return anything while reverseAlt() returns an integer array.
2. reverse() reverses its input in-place while reverseAlt() allocates new memory as large as the original array.
3. reverseAlt() is inefficient because it returns an array, which is a burdensome operation when array is very large.
4. reverse() is better in the sense that it uses significantly less memory space.
5. The following function sorts a given integer array in ascending order. What should be done to reverse the order of sorting?

**static** **void** bubbleSort(**int**[] arr) {

1. **int** temp;
2. **for**(**int** j=0; j<arr.length-2; j++)
3. **for**(**int** i=0; i<arr.length-1-j; i++)
4. **if**(arr[i] > arr[i+1]) {
5. // swap arr[i] and arr[i+1]
6. temp = arr[i];
7. arr[i] = arr[i+1];
8. arr[i+1] = temp;

}

}

1. Replacing the line 2 with **for**(**int** j=arr.length-3; j>=0; j--).
2. Replacing the line 3 with **for**(**int** i=0; i<arr.length-1; i++) and deleting the outer loop.
3. Swapping “arr[i]” and “arr[i+1]” in line 4
4. Swapping “arr[i]” and “arr[i+1]” in lines 6,7,8
5. Write a method which takes an array of strings *dict* and a particular string *pattern* and returns a boolean array having the same length as of *dict* and whose ith element is true iff *pattern* appears somewhere inside *dict[i]*.

1. Write a method which takes an array of strings *dict* and returns a boolean array having the same length as of *dict* and whose ith element is true iff *dict[i]* is a palindrome.
2. Write a function which takes an array of strings and returns the alphabetically first one.

**ANSWERS:**

1. C
2. B
3. A
4. B
5. D
6. A
7. C
8. C
9. **static** **boolean**[] search(String[] dict, String pattern) {

**boolean**[] res = **new** **boolean**[dict.length];

**for**(**int** i=0; i<dict.length; i++)

res[i] = dict[i].contains(pattern);

**return** res;

}

1. **static** **boolean** isPal(String s) {

**if**(s.length() == 1 || s.length() == 0)

**return** **true**;

**else**

**return** s.charAt(0) == s.charAt(s.length()-1) &&

*isPal*(s.substring(1, s.length()-1));

}

**static** **boolean**[] palCheck(String[] dict) {

**boolean**[] res = **new** **boolean**[dict.length];

**for**(**int** i=0; i<dict.length; i++)

res[i] = *isPal*(dict[i]);

**return** res;

}

1. **static** String getFirst(String[] dict) {

String first = dict[0];

**for**(**int** i=0; i<dict.length; i++)

**if**(dict[i].compareTo(first) < 0)

first = dict[i];

**return** first;

}