1. R4.3
   a) 55
   b) 6
   c) 120
   d) 64
2. R4.7
   a) 1 2 3 4 5 6 7 8 9
   b) 1 3 5 7 9
   c) 10 9 8 7 6 5 4 3 2
   d) 0 1 2 3 4 5 6 7 8 9
   e) 1 2 4 8
   f) 2 4 6 8
3. R4.12
   a) java supports while loops, do-while loops, for loops, and for each loops. A while loop is typically used when a sentinel value determines whether it keeps looping. A do-while is the same thing but it always executes the block of code at least once. A for loop is used when you want to make a count controlled loop, and is very useful for iterating through arrays. A for each loop is used to access each element in an array and is useful when you don't need to know about the array index.
4. R4.13
   a) 10
   b) 10
   c) 10
   d) 21
   e) infinite
   f) 11
   g) 7
5. R4.18
   a)
   ```
   int s = 0;
   int i = 1;
   while(i <= 10){
       s += i;
       i++;
   }
   ```
6. R4.21
   a) 2 4 7 11 16
   b) 4 9 16
   c) 10 7
7. R4.28
   a) the largest value could be less than zero or the smallest value could be bigger than zero
8. R4.29
   a) nested loops are a loop inside another loop. They are usually used when traversing 2D arrays.
9. R4.30
   ```
   for(int i = 1; i <= height * width; i++){
       System.out.print("*");
       if(i % width == 0){
           System.out.println();
       }
   ```

```
    }
```
10. R4.31
    a) for hours you would do (int) (Math.random() * 12) + 1
    b) for minutes you would do (int) (Math.random() * 60)
11. R4.32
    a) generate a random number between one and (10 + 3 + 2). If it is between 1 and 10, harry will choose california, if it is between 11 and 13, harry will visit nevada, and if it is 14 or 15, harry will visit utah
12. R6.1
    a)
    ```
    int[] arr = new int[10];
    arr[0] = 17;
    arr[arr.length-1] = 29;
    for(int i = 1; i < arr.length - 1; i++){
        arr[i] = -1;
    }
    for(int val : arr){
        System.out.println(val);
    }
    for(int val : arr){
        System.out.print(val + ", ");
    }
    ```
13. R6.2
    a)
    ```
    for(int i = 0, val = 1; i < 10 && val <= 10; i++, val += 1){
        arr[i] = val;
    }
    ```
    b)
    ```
    for(int i = 0, val = 0; i < 11 && val <= 20; i++, val += 2){
        arr[i] = val;
    }
    ```
    c)
    ```
    for(int i = 0; i < 10; i ++){
        arr[i] = (i + 1) * (i + 1);
    }
    ```
    d)
    ```
    for(int i = 0; i < 10; i ++){
        arr[i] = 0;
    }
    ```
    e)
    ```
    arr[0] = 1;
    arr[1] = 4;
    arr[2] = 9;
    arr[3] = 16;
    arr[4] = 9;
    arr[5] = 7;
    arr[6] = 4;
    arr[7] = 9;
    arr[8] = 11;
    ```
    f)
    ```
    for(int i = 0; i < 10; i++){
        if(i % 2 == 0){
            arr[i] = 0;
        } else {
            arr[i] = 1;
        }
    ```

```
    }
```

g)
```
for(int i = 0; i < 10; i++) {
    if(i > 4) {
        arr[i] = i - 5;
    } else {
        arr[i] = i;
    }
}
```

14. R6.3
    a) 25
    b) 13
    c) 12
    d) index out of bounds
    e) 11
    f) 25
    g) 12
    h) -1
15. R6.4
    a) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
    b) 1, 1, 2, 3, 4, 5, 4, 3, 2, 1
    c) 2, 3, 4, 5, 4, 3, 2, 1, 0, 0
    d) 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    e) 1, 3, 6, 10, 15, 19, 22, 24, 25, 25
    f) 1, 0, 3, 0, 5, 0, 3, 0, 1, 0
    g) 1, 2, 3, 4, 5, 1, 2, 3, 4, 5
    h) 1, 1, 2, 3, 4, 4, 3, 2, 1, 0
16. R6.5
    a)
```
for(int i = 0; i < 10; i++){
    int random = (int)(Math.random() * 100) + 1;
    while(random == values[i]){
        random = (int)(Math.random() * 100) + 1;
    }
    values[i] = random;
}
```
17. R6.6
    a)
```
int[] values = new int[10];
int max = values[0];
int min = values[0];
for(int i = 0; i < values.length; i++){
    if(values[i] < min){
        min = values[i];
    } else if(values[i] > max){
        max = values[i];
    }
}
```
18. R6.7
    a) It will try to access index 10 where 9 is the max
    b) the array was not initialized
19. R6.12

a) An index is a position in the array. The legal indexes are 0 to the length – 1
b) a bounds error is when the programmer tries to access an array index that doesn’t exist

20. R6.14
    a)
```
int[] values = {1, 2, 3, 4, 5, 6, 7, 8, 9};
int[] newValues = new int[values.length];
for(int i = 0; i < values.length; i++){
    newValues[i] = values[i];
}
for(int i = values.length - 1; i >= 0; i--){
    System.out.print(newValues[i]);
}
```
21. R6.15

80 90 100 120 110

pos	found
0	false
1	false
2	True

80 90 100 70

Pos	Found
0	false
1	false
2	true

22. R6.17
    a) public static void sortInDecreasingOrder(int[] array)
    b) public static void printElementsWithSeparator(int[] array, String separator)
    c) public static int countElementsLessThan(int[] array, int threshold)
    d) public static void removeElementsLessThan(int[] array, int threshold)
    e) public static int[] getElementsLessThan(int[] array, int threshold)
23. R6.20
    a) store first element in a temp variable
    b) for each element, assign its own value to the value in the index ahead of it
    c) assign the last element the value of the temp variable
24. R6.22
    a) for each index I, 0 to length – 2
    b) if the new value is <= the value at i and <= the value at I + 1
        1. starting at the end, copy each element to the element in front of it working backwards, ending at I + 1
        2. then assign the new value at position I