**Constructing Vectors with Java**

**(1)** Only a fixed number of elements can be stored in a **Vector**.

(a) True **(b) False**

**(2)** A **Vector** object can shrink during program execution.

**(a) True** (b) False

**(3)** There is a method in the class **Vector** that removes all elements of the **Vector**.

**(a) True** (b) False

**(4)** Every component of a **Vector** object is a reference.

**(a) True** (b) False

**(5)** The class **Vector** is contained in the package **java.swing** .

(a) True **(b) False**

**(6)** Which limitation of arrays does a **Vector** overcome?

**(a) arrays cannot increase in size, vectors can**

(b) arrays cannot be passed as parameters to methods, vectors can

(c) arrays cannot be searched, vectors can

(d) there is a method that returns the length of a vector, there is no way to find the length of an array

**(7)** Which method of the class **Vector** would you use to remove an element at a specific location?

(a) removeAllElements() **(b) removeElementAt()**

(c) removeElement() (d) removeLocation()

**(8)** Which method would you most likely use to find the element in the last location of the vector?

**(a) lastElement()** (b) indexOf()

(c) elementAt() (d) lastIndexOf()

**(9)** Which method would you most likely use to add an element to an end of a vector?

(a) insertElementAt() (b) copyInto()

**(c) addElement**() (d) lastElement()

**(10)** Which package is the class Vector located?

(a) java.io **(b) java.util**

(c) java.lang (d) java.text

**(11)** Which of the following is not characteristic of a vector?

(a) Can grow during program execution

(b) Can shrink during program execution

(c) Can contain items of different types

(d) Can be passed as a parameter to a method

**(e) None of these**

**(12)** Which of the following methods requires the shifting of elements?

(a) replaceAt() (b) insertEnd()

(c) retrieveAt() **(d) insertElementAt()**

**(13)** Which of the following returns the number of elements in an **Vector** named **v** ?

(a) v.length (b) v.numElements()

**(c) v.size()** (d) v.contains

**(14)** What does the following statement do: **Vector thisVector = new Vector();**

**(a) Creates an empty vector**

(b) Creates a vector with 10 elements

(c) This statement does not do anything

(d) Creates an array

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | list | 16 | 30 | 24 | 07 | 25 | 62 | 45 | 05 | 65 | 50 |
|  |  |  |  |  |  |  |  |  |  |  |  |

**(15)** If the list above were implemented as a vector named v1 , how would it be instantiated?

**(a) v1 = new Vector(10)** (b) vector = new v1(10)

(c) v1 = new Vector[10] (d) vector = new v1[10]

**(16)** An array created during the execution of a program is called a(n) \_\_\_\_\_\_\_\_\_\_ array.

(a) static (b) just in time

(c) final **(d) dynamic**

**Programming with Java - Searching and Sorting**

**(17)** The time complexity of the linear search function in an array list is \_\_\_\_\_\_\_\_\_\_ .

(a) *O* ( 1 ) **(b) *O* ( *n* )**

(c) *O* ( 2 ) (d) *O* ( *n* 2 )

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **[6]** | **[7]** | **. . .** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | **list** | **35** | **12** | **27** | **18** | **45** | **16** | **38** | **150** | **. . .** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

**(18)** What is the minimum number of comparisons that have to be made to find 18 using sequential search on the list above?

(a) 1 (b) 3 (c) 2 **(d) 4**

**(19)** What is the minimum number of comparisons that have to be made if the search item was 10 in the list above?

(a) 0 (b) 7 (c) 1 **(d) 8**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **[6]** | **[7]** | **[8]** | **[9]** | **[10]** | **[11]** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **list** | **04** | **08** | **19** | **25** | **34** | **39** | **45** | **48** | **66** | **75** | **89** | **95** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**(20)** What is the minimum number of comparisons that have to be made to find 18 using sequential search on the list above?

(a) 1 (b) 2 (c) 3 (d) 8 **(e) None of these**

**(21)** Using sequential search on the list above, what is the minimum number of comparisons that have to be made if the search item was 10 ?

(a) 0 (b) 1 (c) 7 (d) 10 **(e) None of these**

**(22)** If the list above were to be searched using sequential search, how many key

comparisons would be made to find the number 34 ?

(a) 1 (b) 3 **(c) 5** (d) 12 (e) None of these

**(23)** If the list above were to be searched for the number 34 using binary search, how many key comparisons would have to be made?

(a) 1 (b) 2 (c) 3 **(d) 4** (e) None of these

**(24)** Which element would the search element be compared to first if binary search were used on the list above?

(a) 4 (b) 45 (c) 95 **(d) 39** (e) None of these

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **[6]** | **[7]** | **[8]** | **[9]** |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | **list** | **16** | **30** | **24** | **07** | **25** | **62** | **45** | **05** | **65** | **50** |
|  |  |  |  |  |  |  |  |  |  |  |  |

**(25)** How many key comparisons would have to be made on the list above to find the number 24 ?

(a) 1 (b) 2 **(c) 3** (d) 4 (e) None of these

**(26)** How many key comparisons would have to be made on the list above to find the number 16 ?

**(a) 1** (b) 2 (c) 3 (d) 4 (e) None of these

**(27)** How many key comparisons would have to be made on the list above to find the number 5 ?

(a) 4 (b) 5 (c) 7 **(d) 8** (e) None of these

**(28)** On average, how many comparisons would have to be made to find an element in the list above?

(a) 2 **(b) 5** (c) 6 (d) 8 (e) None of these

**(29)** If the list above were sequentially sorted, which two elements would be swapped first?

(a) 5 and 16 (b) 5 and 45

(c) 65 and 16 **(d) 24 and 30**

**(30)** Why can the binary search method not be used on the list as it appears above?

(a) Because the list is too big **(b) Because the list is not sorted**

(c) Because it is a list of integers (d) Because it is an array

**(31)** Which method does binary search use to find an element in a list?

**(a) divide and conquer** (b) linear search

(c) row and column (d) hunt and peck

**(32)** What is usually returned if the search item is found during a search of a list?

**(a) the location of the element** (b) − 1

(c) the element (d) true

**(33)** In the binary search algorithm, each time through the loop we make \_\_\_\_\_\_\_\_\_\_ key comparison(s).

**(a) one** (b) two (c) three (d) four (e) None of these

**(34)** Unordered arrays may be searched more efficiently than ordered arrays.

(a) True **(b) False**

**(35)** Consolidating two ordered arrays into a third ordered array is called sorting the arrays.

(a) True **(b) False**

**(36** An entire local array can be passed to another procedure.

**(a) True** (b) False

**(37)** The bubble sort is an algorithm that compares adjacent items and swaps those that are out of order.

**(a) True** (b) False

**(38)** Java allows the programmer to define one, two, three ( or higher ) dimensional arrays.

**(a) True** (b) False

**(39)** All arrays in Java have \_\_\_\_\_\_\_\_\_\_ for the lower - bound value.

(a) − 1 **(b) 0** (c) 1 (d) 10 (e) 20

**(40)** If an array is to have five elements, you declare it with an upper - bound value of \_\_\_\_ .

(a) 0 **(b) 4** (c) 5 (d) 6 (e) 10

**(41)** For the array **{0, 55, 33, 12}** the upper bound of the array is \_\_\_\_\_\_\_\_\_\_ .

**(a) 3** (b) 2 (c) 12 (d) 0 (e) 4

**(42)** Using the array below and assuming the bubble sort is used to sort the array **Age** from smallest to largest, what are the contents of the array after three passes?

**Age: 17 , 8 , 21 , 47 , 5 , 17 , 31**

(a) 8 , 5 , 21 , 47 , 17 , 17 , 31

(b) 5 , 8 , 17 , 17 , 21 , 47 , 31

(c) 5 , 8 , 17 , 21 , 17 , 31 , 47

(d) 8 , 5 , 17 , 21 , 17 , 47 , 31

**(e) None of these**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 08 | 17 | 05 | 17 | 21 | 31 | 47 | third pass |
|  |  |  |  |  |  |  | fourth comparison |

**(43)** Which of the following arrays could be searched most efficiently?

(a) years = {4301, 1944, 2011, 4445}

(b) cities = {"Sam", "Pete", "Gary", "James"}

**(c) neighborhoods = {"A", "C", "C", "K"}**

(d) number = {349, 347, 347, 349}

**(44)** In the ordered array declared with the statement below, the value of the element men[1] is equal to that of which of the other following elements?

**men = {"", "jim", "jim", "james", "james"}**

(a) men[0] (b) men[1] **(c) men[2]** (d) men[3] (e) men[4]