Unit 4 – ICS4U – Arrays & Algorithms

Sample Test, November 8, 2022

Name: Solutions

Total	%	Knowledge	Communication	Thinking	Application
(108)	%	(24)	(24)	(31)	(29)

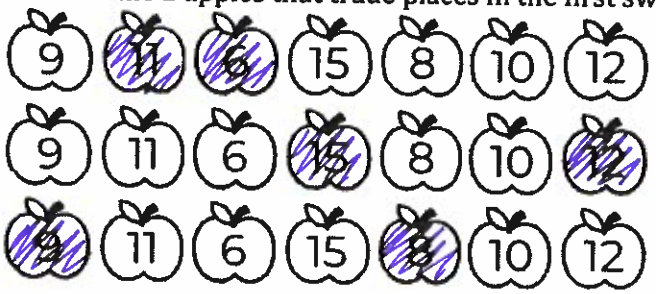
Knowledge

1. For each sorting algorithm, colour in the 2 apples that trade places in the first swap. /3

(a) Bubble sort

(b) Selection sort

(c) Quicksort



2. Sort the array using each algorithm. /10

Bubble sort:

8	7	6	5	4
7	8	6	5	4
7	6	8	5	4
7	6	5	8	4
7	6	5	4	8
6	7	5	4	8
6	5	7	4	8
6	5	4	7	8
5	6	4	7	8
5	4	6	7	8
4	5	6	7	8

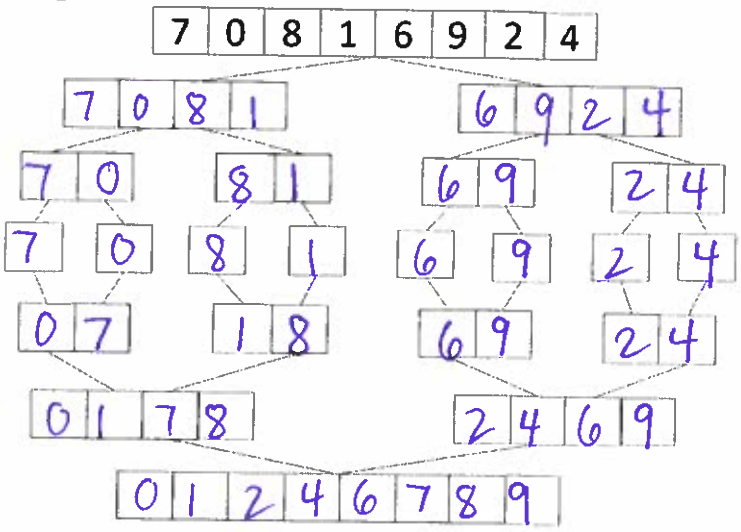
Selection Sort:

7	3	9	6	8	5	4
7	3	4	6	8	5	9
7	3	4	6	5	8	9
5	3	4	6	7	8	9
4	3	5	6	7	8	9
3	4	5	6	7	8	9

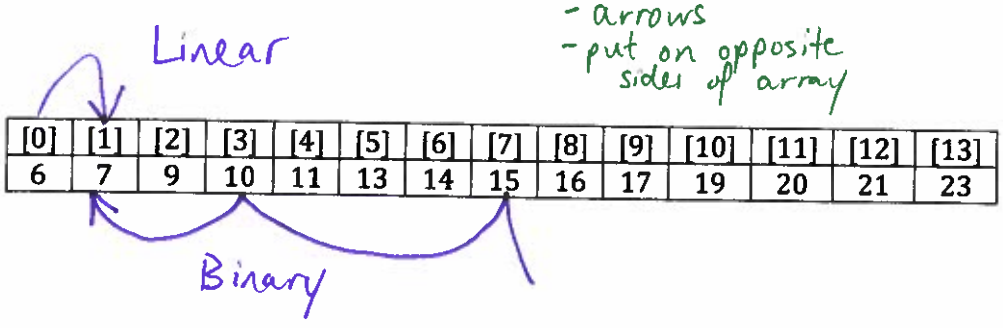
Quick Sort: (1ST pass only)

4	0	6	5	7	8	3	2
2	0	6	5	7	8	3	4
2	0	4	5	7	8	3	6
2	0	3	5	7	8	4	6
2	0	3	4	7	8	5	6

Mergesort:



3. Trace the search of this array to find '7' using BOTH linear search and binary search.
When drawing on the array, label the binary search and label the linear search. /6



For Binary Search:

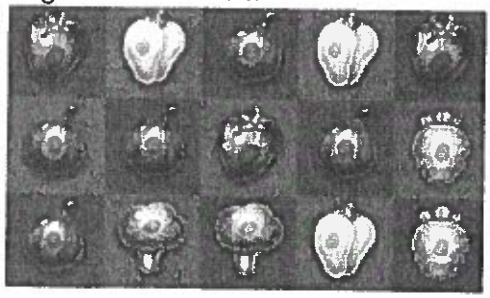
Low	High	Mid
0	14	7
0	6	3
0	2	1

4. Using the pictures shown, fill in the code for this game of Farm Swap. /5

The picture information:

1 onion 2 tomato 3 broc 4 pepper 5 carrot

The grid information:



The code:

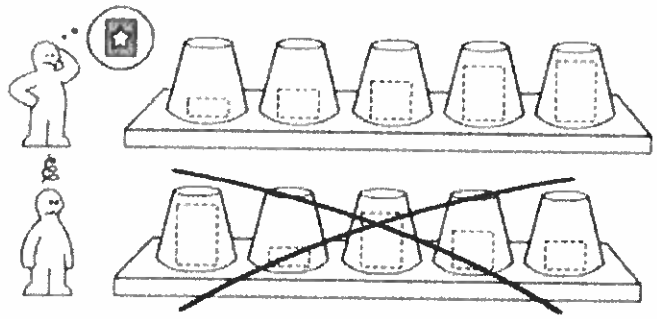
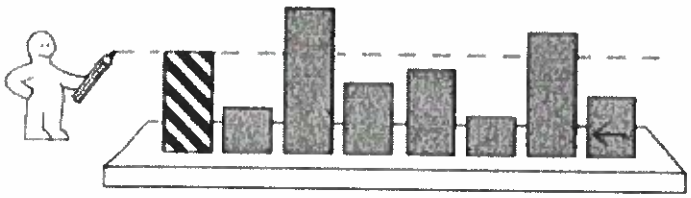
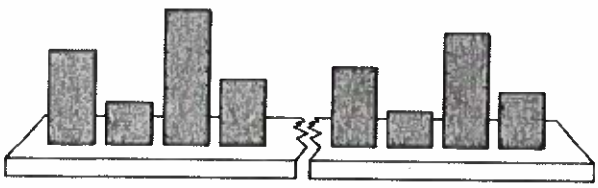
```
int field [][]={{2, 4, 1, 4, 2},
                {1, 1, 2, 1, 5},
                {1, 3, 3, 4, 5}};
int row = 3;
int col = 5;
JButton pics[]=new JButton[row*col];

Panel grid=new Panel(new GridLayout(row, col));
int m=0;

for(int i=0; i<row; i++){
   for(int j=0; j<col; j++){
       pics[m]=new JButton
        (createImageIcon(field[i][j]+".jpg"));
       pics[m].addActionListener(this);
       pics[m].setActionCommand(m +"");
       grid.add(pics[m]);
       m++;
   }
}
add(grid);
```

Communication

5. Each of the following images were taken from a pictorial representation of an algorithm. Identify the algorithm shown and explain your choice. /6

Image	Which Algorithm?	Why that Algorithm?
	Binary Search	- character is thinking of thing to find - algorithm works with sorted data (top row) but not with unsorted (bottom row)
	Quick Sort	- first element is special ... it is the pivot. - all others compared to pivot.
	Merge Sort	- breaking in half - as "division" phase of mergesort

6. Provide the phrase or term indicated. /10

pivot	(a) In quicksort, a partition is finished when this is in its correct place.
John von Neumann	(b) He wrote Mergesort.
merge	(c) The second method, besides divide, used by Mergesort.
Bubblesort	(d) The first sorting algorithm coded.
Number of elements in array	(e) The 'n' in Big-Oh notation stands for this.
Quicksort	(f) Tony Hoare won the Turing award for this algorithm in 1980.
Jon Bentley	(g) Who proved that fast hardware can not compensate for a slow algorithm?
Bogosort	(h) The absolute worst sorting algorithm.
Swaps	(i) Quicksort does this much more purposefully than Bubblesort.
Selection sort	(j) The sorting algorithm for how children eat Halloween candy.
Search	(k) Finding the location of an item in an array.

7. Describe the trade-offs for each of the following things. Use one sentence for the positive and another for the negative. Be careful there are two sentences to fully explain your ideas. /8

(a) Quicksort

Positive: It is the fastest in-place algorithm in the general case. Really fast; works in O(n log n) speed.

Negative: Doesn't work well for non-random data, say reverse order or almost sorted.

(b) Bubble sort

Positive: It's best case, the array is almost sorted, is very fast. In this case, it's speed is close to O(n).

Negative: All other cases are very slow. It's many, many, many swaps are inefficient.

(c) Binary Search

Positive: It is much faster than Linear search. Linear is O(n) speed while Binary is O(log n).

Negative: It uses the sorted structure of data to gain it's speed. Sorting is a slow operation.

(d) Selection sort

Positive: It is easy to understand & code. It is based on the max function.

Negative: It's simplicity makes it efficient. Selection Sort is slow: $O(n^2)$.

Thinking

row/x = n/6
col/y = n%6

8. Assume that you have a grid that is 9 (rows) x 6 (cols). /5

(a) How many ImageViews do you need? . 54...

(b) Given the ImageView IDs, determine each button's (x, y), aka (row, col), position in the int tracking array.

8 1,2	10 1,4	36 6,0	42 7,0

9. Which **sorting** algorithm is the best choice for each situation? /6

Bubble (a) The list of names is sorted, you add one element to the front. You have extra memory.

Selection (b) The array not randomized, it is very large. You just have enough memory to hold it.

Merge (c) The char array is in reverse order, but it is only 62 elements long.

Quick (d) You have lots of extra memory and the double array is in random order.

Quick (e) You have a list of the first 10 million digits of PI. You have no extra memory.

Bubble (f) Your phone book is sorted, you add one new person to the book. Now it is almost sorted.

10. In each case, which **search** would be the best choice? /5

Binary (a) You have a list of heights, ordered from smallest to greatest.

Binary (b) You are looking up a word in a dictionary.

Linear (c) You are looking up a definition in a dictionary to find the word that goes with it.

Linear (d) You are looking up a student ID in a list with no apparent order.

Linear (e) You are looking up a student ID in a list ordered by student last name.

11. Put these algorithm speeds in order. (1 is fastest, 6 is slowest)

3 O(n)
6 O(n!)
2 O(log n)
1 Constant time
4 O(n log n)
5 O(n²)

12. What speeds are these array algorithms? /10

(a) Swap O(1)
(b) Quicksort O(n log n)
(c) Binary Search O(log n)
(d) Bubble sort (average case) O(n²)
(e) BogoSort O(n!)
(f) Linear Search O(n)

13. Circle **and correct** 5 errors in the following two methods needed for BogoSort. /5

```
public void bogoSort (int a[]) {
   while (!isSorted (a)) {

     for (int i = 0 ; i < 60 ; i++) {
        int b = (int) (Math.random()*a.length);
        int c = (int) (Math.random()*a.length);
        int temp = a [b];
        a [b] = a [c];
        a [c] = a[b];
     }

     for (int i = 0 ; i < a.length;
         System.out.print (a [i] + " ");
      System.out.println ();
   }
}
```

; 1
temp 2
i++ 3

```
public double isSorted (int a[]) {
   for (int i = 0 ; i < a.length-1 ; i++) {

      if (a [i] > a [i + 1])
          return false;
      }

      return true;
   }
}
```

boolean 4
5

Application

14. Convert each method so it works with a String array. /8

```
public void unsort(String a[]) {

   for (int i = 0; i < 60; i++) {
      int c = (int) (Math.random() * a.length);
      int b = (int) (Math.random() * a.length);
      String temp = a[c];
      a[c] = a[b];
      a[b] = temp;
   }
}
```

```
public int binary (String a[], String x, int low, int high) {

    if (low > high) return -1;
    int mid = (low + high)/2;
    if (a[mid].equals(x)) return mid;
    else if (a[mid].compareTo(x) < 0)
       return binary(a, x, mid+1, high);

    else
       return binary(a, x, low, mid-1);
}
```

```
public void bubbleSort(String a[]) {
  int n = a.length;
  for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - 1 - i; j++) {
      if (a[j+1].compareTo(a[j]) < 0) {
        String temp = a[j];
        a[j] = a[j + 1];
        a[j + 1] = temp;
      }
    }
  }
}
```

15. Write the java method for the Farm Swap game that assigns a carrot (5) to each element in the `field` array. Its name is `carrotsEveryWhere`. The variables `row` and `col` track the dimensions for the array. Make sure that the new array is updated on the screen using `redraw`. The method has no parameters, and returns nothing.

/7

```
public void carrotsEveryWhere() {
    for(int i=0; i<row; i++) {
        for(int j=0; j<col; j++) {
            field[i][j] = 5;
        }
    }
    redraw();
}
```

16. Fill in the neighbours chart for the center element. /5

row one less

Same row

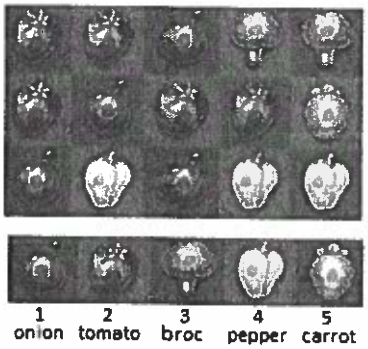
row one more

Up-Left	Up	Up-Right
field[x-1][y-1]	field[x-1][y]	field[x-1][y+1]
Left	**Clicked Element**	**Right**
field[x][y-1]	field[x][y]	field[x][y+1]
Down-Left	**Down**	**Down-Right**
field[x+1][y-1]	field[x+1][y]	field[x+1][y+1]

Column one less — same column — Column one more

17. In the game Farm Swap, you get extra points every time you get an onion (1) under a tomato (2). In the screen shown, the user would get three extra points.

Code a method that returns the number of onions under tomatoes in an global array named `field`. The method has no parameters, but will return an integer.



/9

```
public int extraPoints() {
    int total = 0;
    for(int i=0; i<row; i++) {
        for (int j=0; j<col; j++) {
            if ( i-1>=0 && field[i-1][j]==2 && field[i][j]==1 )
                total += 3;
        }
    }
    return total;
}
```

edge guard — tomato above — onion below