**Module 1: Cairn**

**Question 1 0.5 pts**

Suppose that you had a program written in Java that maintains a list of courses that a student was taking, and that program was exhibiting issues from reference types. Which of the following would most likely indicate that issue?

1. The program crashes immediately with a null pointer exception.
2. The program prompts the user for input, but instead of waiting, instantly reads a null string and continues.
3. The program crashes immediately with an error message about allocating memory for a new Student object.
4. Even after adding ten unique classes, when displaying everything, it shows the same ten classes.

***Answer:*** 4

**Question 2 1.5 pts**

Which of the following statements are true of the concepts listed below?

A) Defines operations.

B) Defines state.

C) Not concerned with internal implementations.

1. A) is true of ADTs.
2. B) is true of ADTs.
3. C) is true of ADTs.
4. A) is true of Classes.
5. B) is true of Classes.
6. C) is true of Classes.
7. A) is true of Interfaces.
8. B) is true of Interfaces.
9. C) is true of Interfaces.

***Answer:*** 1,3,4,5,7,9

**Question 3 2 pts**

Given the following API for a mutable Employee ADT, determine if each of the operations is an extraction, or transformation.

int getId(); [Extraction / Transformation]

void logWorkHours(double hours); [Extraction / Transformation]

double calculatePayCheck(); [Extraction / Transformation]

void logVacationTime(int days); [Extraction / Transformation]

int vacationTimeAvailable(); [Extraction / Transformation]

void logSickDays(int days); [Extraction / Transformation]

int paidSickDaysAvailable(); [Extraction / Transformation]

***Answer:***

1. Extraction
2. Transformation
3. Extraction
4. Transformation
5. Extraction
6. Transformation
7. Extraction

**Question 4 2 pts**

The following class represents a 3d shape using an array of 3dPoints. It is intended to be immutable. Assume the 3dPoint class is immutable.

Some of these functions, by their nature, make the class mutable and should be removed completely to make it immutable.

Some others might make the class mutable, but it is possible to make them safe by implementing them carefully.

Others are mostly safe in an immutable class.

Determine which functions should be removed, which must be carefully implemented, and which are safe.

class 3dShape {  
 private 3dPoint points;  
 public 3dShape(3dPoint[] points);  
   
 *// Sets the point array member to the given point array* public void setPoints(3dPoint[] points);  
   
 *// Returns the point array member* public 3dPoint[] getPoints();  
   
 *// Returns the volume of the shape represented by the point* public double getVolume();  
   
 *// Stretches the shape by the percent specified in each direction* public void stretch(double x, double y, double z);  
}

Choices are:

* needs to be removed.
* needs to be carefully implemented.
* does not need special attention.

function 1:

function 2:

function 3:

function 4:

***Answer:***

function 1: needs to be removed.

function 2: needs to be carefully implemented.

function 3: does not need special attention.

function 4: needs to be removed.

**Question 5 1 pts**

Select the functions from the following list that could belong to an ADT. Remember that not all ADTs are immutable. (Hint: ask yourself if these methods belong in an idealistic ADT, i.e. one truly meeting the ADT design rule.)

1. Account ADT

// Accepts an int and sets the account’s opening year to that

setYear(int year)

1. Bag ADT

// Sets the max capacity of the bag to the given integer

setMax(int capacity)

1. Queue ADT

// Sets the queues internal ArrayList to the given ArrayList

setMaxCapacity(int capacity)

1. Queue ADT

// Removes the least recently added item

dequeue() throws ArrayOutOfBoundsException

***Answer:***

1,2

**Module 2: Cairn**

**Question 1 0.5 pts**

Consider the following implementation of a 1D vector class:

class IntVector {

private final int[] data;

public IntVector(int[] vector) {

if (vector == null)

throw new IllegalArgumentException();

data[y] = vector[y];

}

// ...

}

Is this class immutable? Answer yes/no, and justify.

1. Yes - the constructor validates that there is no outside refeference to the data and throws an exception if there is.
2. Yes - the member variable is set to final and the constructor makes a deep copy.
3. No - although the member variable is set to final, the constructor does not make a deep copy.
4. No - the constructor always throws an exception before it is able to copy the data.
5. No - this would not even compile since there is an assignment to a final variable in the constructor.
6. Yes - the member variable is set to final and that is sufficient.

***Answer:*** 5.

**Question 2 1 pts**

Select the functions from the following list that could belong to an ADT. (Hint: ask yourself if these methods belong in an idealistic ADT, i.e. one truly meeting the ADT design rule.)

1. Stack ADT

// add an item

push(Item item)

1. Stack ADT

// Sets the stack's internal ArrayList to the given ArrayList

setContents(ArrayList contents)

1. Stack ADT

// Sets resize factor for the internal array

setResizeFactor(float rsf)

1. Stack ADT

// remove the most recently added item

pop() throws EmptyStackException

1. Stack ADT

// remove the most recently added item

pop() throws ArrayOutOfBoundsException

***Answer:*** 1,4.

**Question 3 1 pts**

It is correct to say that the nodes in a linked list are... (select all that apply)

1. statically allocated.
2. recursive data.
3. dynamically allocated.
4. contiguously allocated next to each other in main memory

***Answer:*** 2,3.

**Question 4 0.5 pts**

Assume that head is the head node of a singly-linked list containing the nodes A, B, and C. What would the result of executing the following code be? Indicate the resulting list using box and arrow notation.

head.setNext(head.getNext().getNext());

1. A null pointer exception will be thrown.
2. A blue rectangular with white text

   Description automatically generated
3. A blue rectangle with white text

   Description automatically generated
4. None of these.
5. A blue rectangle with white text

   Description automatically generated

***Answer:*** 5.

**Question 5 0.5 pts**

Consider the code below. Assume head points to the beginning of a list and is a class variable

public void methodA(Node newNode) {  
 if(head == null)  
 head = newNode;  
 else {  
 newNode.next = head;  
 head = newNode;  
 }  
}

public void methodB(Node newNode) {  
 Node iter = head;  
 if(head == null)  
 head = newNode;  
 else {  
 while(iter.next != null)  
 iter = iter.next;  
 Iter.next = newNode;  
 }  
}

Suppose the list already has five elements. What code would be faster in terms of adding an element to the list?

1. Both are the same because they both add a node to a linked list.
2. Method A because you don’t have to traverse through the list.
3. Method B because it checks null before spending time to do anything else.
4. Method A because it has fewer lines of code.
5. Method B because it safely adds a new node to the list using an iterator node.

***Answer:*** 2.

**Question 6 2 pts**

Assume we have an array with contents 1, 5, 3, 7, 8. Those contents could also be stored with a singly linked list (with 1 at the head, no tail variable). When considering performance:

It is faster to access 8 in a [array].

It is faster to access 1 in a [array].

Resizing a linked list would typically be [faster] as compared to an array because a linked list is [dynamically allocated].

**Question 7 0.5 pts**

Compare the amount of memory needed to store an array of 100 integers and a linked list containing 100 integers. Which requires less? Or are they the same? Justify.

1. List - it will be exactly the number of elements until the array.
2. Array - since that will allow the use of bracket notation.
3. Array - it will only store the integers, no other references.
4. List - they are always smaller than arrays.
5. Same - they store the same number of elements.

***Answer:*** 3.

**Question 8 1 pts**

Which of the following statements are true about generics? Select all that apply.

1. They allow for different data types in the code without specifying one at the beginning.
2. They are limited to object types but can still use primitives through wrapper classes.
3. They make compiled files smaller.
4. They make code reusable.

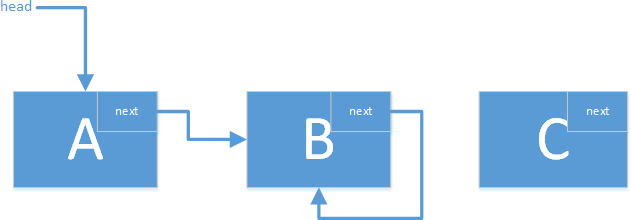
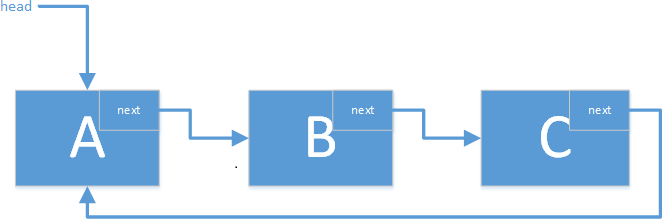
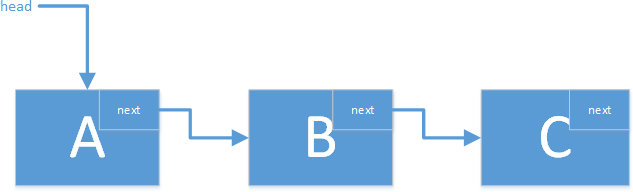
***Answer:*** 2.

**Module 3: Cairn**

**Question 1 1 pts**

[Acuña] Assume that head is the head node of a singly linked list containing the nodes A, B, and C. What would the result of executing the following code be? Indicate the resulting list using box and arrow notation.

head.getNext().setNext(head.getNext());

1. None of these.
2. A null pointer exception will be thrown.
3. 
4. 
5. 

***Answer:*** 3.

**Question 2 1 pts**

Suppose that you are asked to analyze the Big-Oh of the following method pop(). What pieces of additional information which need to give a perfectly accurate answer? Select all that apply.

public ItemType pop() {

if(isEmpty())

throw new NoSuchElementException();

ItemType element = top.getElement();

top = top.getNext();

n--;

return element;

}

1. The Big-Oh of the constructor of the NoSuchElementException class.
2. The number of times pop() will be called.
3. The type that will be used in place of ItemType.
4. The Big-Oh of isEmpty().

***Answer***

2,4

**Question 3 1 pts**

Assuming there is a growth function , what does the value of represent in terms of an algorithm?

***Answer***

helps us understand the baseline or "fixed" cost of an algorithm when the input size is minimal or zero.

**Question 4 0.5 pts**

What is the Big-Oh order of the following growth expression?

***Answer***

**Question 5 0.5 pts**

Is it *correct* to say that the expression in the previous question is ? Explain.

***Answer***

No; because:

* is **polynomial growth** with a logarithmic factor.
* is **exponential growth**, which increases much faster than any polynomial or logarithmic function.

**Question 6 0.5 pts**

What is the Big-Oh order of the following code fragment? The size of the problem is expressed as n.

for (int i = 1; i <= n; i++)

  for (int j = 1; j <= n; j += 2)

   System.out.println("Nested loops!");  //f(n) counts these

***Answer***

**Question 7 0.5 pts**

What is the Big-Oh order of the following code fragment? The size of the problem is expressed as n.

for (int i = 0; i < (int)Math.pow(2, n); i++)

System.out.println("what could could go wrong?"); //f(n) counts these

***Answer***

**Question 8 0.5 pts**

What is the tilde approximation of the following growth function?

1. Does not exist.

***Answer***

**Question 9 0.5 pts**

You are in the process of choosing between two 3rd party libraries that implement some algorithm and have found two solutions, A and B, that are advertised as and , respectively. Which of these solutions would you prefer in terms of performance? Explain. (Hint: think about which one gives a more "stable" result).

***Answer***

In terms of **performance**, you would prefer **Solution B** , as it provides more precise and predictable performance characteristics. The tilde notation suggests that the algorithm's behavior is well-analyzed and predictable, which often translates to better and more stable real-world performance.

**Module 5: Cairn**

**Question 1 1 pts**

Consider the following problem, and categorize according to the different axis of problem complexity: create an algorithm to determine which song from a collection to recommend to a user, based on what they have listened to in the past.

It is    **[ Select ]**   ["open-ended", "close-ended"]  and     **[ Select ]**   ["ill-defined", "well-defined"].

***Answer:***

Open-ended and ill-defined.

**Question 2 1 pts**

Understanding the data format and ordering requirements for an algorithm that needs to (as a sub-problem) sort data would take place during         **[ Select ]**      ["Analysis", "Design", "Justification"]  , while analyzing the sorting algorithm's ability to meet those requirements would happen during         **[ Select ]**      ["Analysis", "Justification", "Design"]  .

***Answer:***

Design, analysis.

**Question 3 1.5 pts**

Consider the following array: 21,16,3,7,23,12. Show a trace of execution for**selection sort.** The trace begins with the provided initial state of the array, followed by the array's state after each **swap** is made.

*Enter each subsequent state as a comma separated list (as shown by the initial state). Do not include any spaces. If you follow these directions exactly and are marked off by the auto-grader on the second submission, reach out to the instructional staff.*

initial: 21,16,3,7,23,12

i  =  0: 3,16,21,7,23,12

i  =  1:

i  =  2:

i  =  3:

i  =  4:

i  =  5:

***Answer***

i = 1: 3,7,21,16,23,12

i = 2: 3,7,12,16,23,21

i = 3: 3,7,12,16,23,21

i = 4: 3,7,12,16,21,23

i = 5: 3,7,12,16,21,23

**Question 4 1.5 pts**

Consider the following array: 21,16,3,7,23,12. Show a trace of execution for **insertion sort**. The trace begins with the provided initial state of the array, followed by the array's state after each **pass** is made.

*Enter each subsequent state as a comma separated list (as shown by the initial state). Do not include any spaces in your answer. If you follow these directions exactly and are marked off by the auto-grader on the second submission, reach out to the instructional staff.*

initial: 21,16,3,7,23,12

i  =  1: 16,21,3,7,23,12

i  =  2:

i  =  3:

i  =  4:

i  =  5:

***Answer***

i = 2: 3,16,21,7,23,12

i = 3: 3,7,16,21,23,12

i = 4: 3,7,16,21,23,12

i = 5: 3,7,12,16,21,23

**Question 5 1 pts**

Selection sort assumes there is a region of         [ Select ]      ["unsorted elements at the front", "sorted elements at the front", "sorted elements at the back", "unsorted elements at the back"]  of a collection, picks the first unsorted element and places it         [ Select ]      ["in the middle of", "at the beginning", "at the end"]  of the sorted region, until the entire list is sorted.

***Answer:***

sorted elements at the front, at the end.

**Question 6 0.5 pts**

What is the Big-Oh order of the following code fragment? The fragment is parametrized on the variable N. Assume that you are measuring the number of assignments to min.

int N = a.length;

for (int i = 0; i < N; i++) {

int min = i; //assignment to min

for (int j = i+1; j < N; j++)

if (less(a[j], a[min]))

min = j; //assignment to min

exch(a, i, min);

}

Group of answer choices

1. Does not exist, or cannot be determined with information given.

***Answer***

**Question 7 0.5 pts**

What is the most efficient sorting algorithm to use for the data set 1,3,5,7,9 assuming you want to sort the numbers in ascending order.

Group of answer choices

1. Bogo Sort.
2. Shell Sort.
3. Selection Sort.
4. Insertion Sort.

***Answer***

Insertion Sort

**Module 6: Cairn**

**Question 1 1 pts**

Consider the following array: 3,8,23,18,15,16. Show a trace of execution for **insertion sort**. The trace begins with the provided initial state of the array, followed by the array's state after each **pass** is made.

*Enter each subsequent state as a comma separated list (as shown by the initial state). Do not include any spaces in your answer. If you follow these directions exactly and are marked off by the auto-grader on the second submission, reach out to the instructional staff.*

initial: 3,8,23,18,15,16

i  =  1: 3,8,23,18,15,16

i  =  2:

i  =  3:

i  =  4:

i  =  5:

***Answer***

i  =  2: 3,8,23,18,15,16

i  =  3: 3,8,18,23,15,16

i  =  4: 3,8,15,18,23,16

i  =  5: 3,8,15,16,18,23

**Question 2 0.5 pts**

Match the following scenarios with which sorting algorithm is potentially most appropriate. (Find the best matching, with using each algorithm exactly once.)

1. Datasets that are already almost sorted.

           [ Choose ]             Shellsort             Selection Sort             Mergesort             Insertion Sort

1. Datasets being analyzed on system with a limited number of writes.

           [ Choose ]             Shellsort             Selection Sort             Mergesort             Insertion Sort

1. Large datasets where memory is not a limitation.

           [ Choose ]             Shellsort             Selection Sort             Mergesort             Insertion Sort

1. Datasets where elements are very far away from where they need to be.

           [ Choose ]             Shellsort             Selection Sort             Mergesort             Insertion Sort

***Answer***

1. Insertion Sort
2. Selection Sort
3. Mergesort
4. Shellsort

**Question 3 3 pts**

Consider the following array: 2, 13, 16, 3, 7, 23, 12, 25. Show a trace of execution for top-down mergesort using the method shown in lecture (where both sides are updated at once). Illustrate the contents of the array as it is broken down, and then merged into an ordered state.

*Enter each subsequent state as a comma separated list (as shown by the initial state). Do not include any spaces or brackets in your answer. If you follow these directions exactly and are marked off by the auto-grader, reach out to the instructional staff.*

1 (initial) : 2,13,16,3,7,23,12,25 (given as example)

2 (down) : 2,13,16,3,7,23,12,25 (given as example; nothing changed!)

3 (down) :

4 (mid) :

5 (up) :

6 (up) :

7 (up) :

***Answer***

3 (down) : 2,13,16,3,7,23,12,25

4 (mid) : 2,13,16,3,7,23,12,25

5 (up) : 2,13,3,16,7,23,12,25

6 (up) : 2,3,13,16,7,12,23,25

7 (up) : 2,3,7,12,13,16,23,25

**Question 4 0.5 pts**

What is the Tilde approximation order of the following code fragment? Assume that you are measuring the number of assignments to a[], and that n represents the number of indices between lo and hi, inclusive.

public static void merge(Comparable[] a, Comparable[] aux, int lo, int mid, int hi) {

int i = lo, j = mid+1;

for (int k = lo; k <= hi; k++)

aux[k] = a[k];

for (int k = lo; k <= hi; k++) {

if (i > mid) a[k] = aux[j++];

else if (j > hi) a[k] = aux[i++];

else if (less(aux[j], aux[i])) a[k] = aux[j++];

else a[k] = aux[i++];

}

}

1. Does not exist or cannot be determined with information given.

***Answer:*** 2

**Question 5 0.5 pts**

If you need to sort a large dataset on a system with limited memory, would it be a good idea to use merge sort? Explain.

1. Yes - merge sort has a cool name, so it's guaranteed to run fast.
2. No - merge sort requires a linear number of recursive calls to be made.
3. Yes - merge sort has proven optimal performance.
4. No - merge sort requires an auxiliary array to do the merging step.

***Answer***

4

**Question 6 0.5 pts**

When we talked about insertion sort, we found that it requires only comparisons and 0 exchanges when the input is sorted. Does this violate the lower bound proof?

1. Yes - this means that some inputs will become unsorted.
2. No - the lower bound proof only applies to merge sort or other merging sorting algorithms.
3. Yes - no sorting algorithm can ever do better than comparisons.
4. No - a sorted input is a special case, while the lower bound proof applies to all inputs.

***Answer***

4

**Module 7: Cairn**

**Question 1 1.5 pts**

Consider the following array: 8,9,17,4,3,20,25,5 Show a trace of execution for top-down mergesort using the method shown in lecture (where both sides are updated at once). Illustrate the contents of the array as it is broken down, and then merged into an ordered state.

*Enter each subsequent state as a comma separated list (as shown by the initial state). Do not include any spaces or brackets in your answer. If you follow these directions exactly and are marked off by the auto-grader, reach out to the instructional staff.*

1 (initial) : 8,9,17,4,3,20,25,5 (given as example)

2 (down) : 8,9,17,4,3,20,25,5 (given as example; nothing changed!)

3 (down) :

4 (mid) :

5 (up) :

6 (up) :

7 (up) :

***Answer***

3 (down) : 8,9,17,4,3,20,25,5

4 (mid) : 8,9,17,4,3,20,25,5

5 (up) : 8,9,4,17,3,20,5,25

6 (up) : 4,8,9,17,3,5,20,25

7 (up) : 3,4,5,8,9,17,20,25

**Question 2 0.5 pts**

How (in general) are priority queues related to other sorting algorithms?

Group of answer choices

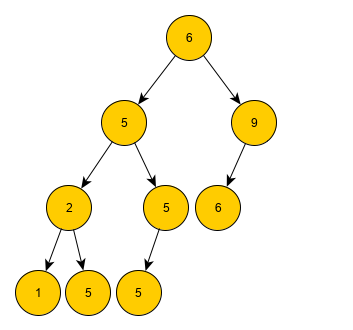
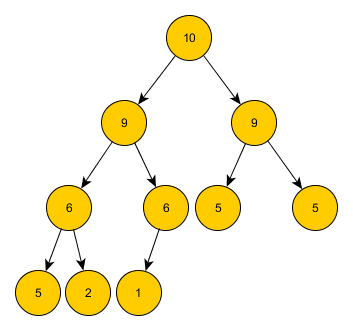
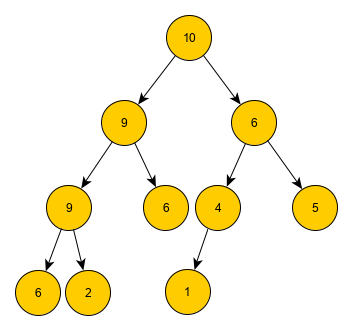
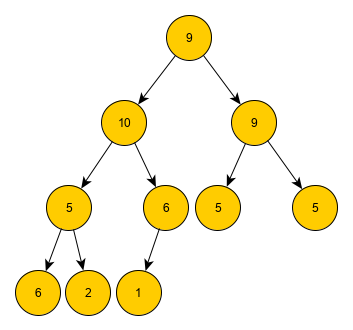
1. Both priority queues and the other four sorting algorithms must obey the lower bound proof.
2. They both involve comparing elements, and PQs may be used to create a straightforward sorting algorithm.
3. Priority queues are implementations of the same ADT that the sorting algorithms use.
4. Priority queues are more efficient implementation of insertion sort.

***Answer***

They both involve comparing elements, and PQs may be used to create a straightforward sorting algorithm.

**Question 3 1.5 pts**

Consider the following values: 5, 9, 1, 5, 6, 2, 5, 6, 9, 10. Select a valid binary heap for this data.

1. 
2. 
3. 
4. 

***Answer:*** 2

**Question 4 1 pts**

Is your answer to the previous question unique? Explain.

Group of answer choices

1. No - the nodes can also be arranged to fit the BST ordering rule.
2. Yes - these nodes can only be used to draw a specific tree.
3. No - the same nodes can be used to draw multiple valid heaps.
4. Yes - the previous question had only one answer.

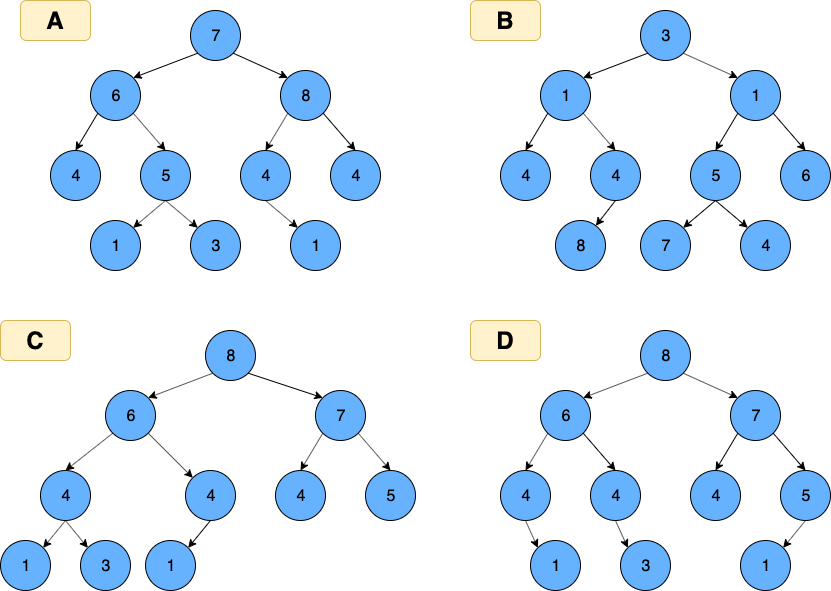
***Answer***

No - the same nodes can be used to draw multiple valid heaps.

**Question 5 1.5 pts**

Consider the following priority queue array: [0, 8, 6, 7, 4, 4, 4, 5, 1, 3, 1]. Which of the diagrams below accurately represents it?

Remember: The first element of the array is never used.

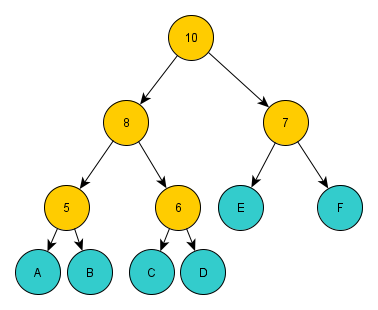


1. A
2. C
3. D
4. B

***Answer:*** C

**Question 6 0.5 pts**

Suppose that you are inserting a node with value 4 into the max PQ show below. When it is initially added, which position (see lettered nodes) does it have in the heap/tree?



1. A
2. C
3. F
4. E
5. B
6. D

***Answer***

6

**Question 7 0.5 pts**

What is the Tilde approximation order of the following code fragment? Assume that you are measuring the number of comparisons (*less*) and that n represents the number of nodes in the heap.

private void sink(int k) {

while (2\*k <= N) {

int j = 2\*k;

if (j < N && less(j, j+1)) j++;

if (!less(k, j)) break;

exch(k, j);

k = j;

}

}

1. Does not exist, or cannot be determined with information given.

***Answer***

**Quiz Midterm**

**Question 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | In a stack, if a user tries to remove an element from an empty stack it is called... | | | | Answer:  a |
| a. Underflow. | b. Empty collection. | c. Overflow. | d. Garbage collection. |
|  | Pushing an element into stack already having five elements and stack size of 5, then stack becomes... | | | | Answer:  a |
| a. Overflow. | b. Crash. | c. Underflow. | d. User flow. |
|  | A linear collection of data elements where the linear node is given by means of pointer is called? | | | | Answer:  b |
| a. Node list. | b. Linked list. | c. Primitive list. | d. Unordered list. |
|  | In a generic stack implementation, what does the term "generic" mean? | | | | Answer:  c |
| a. The stack can only store integers. | b. The stack can only store strings. | c. The stack can store elements of any data type. | d. The stack can store elements of a specific data type chosen at runtime. |
|  | Linked list is considered as an example of a type of memory allocation... | | | | Answer:  a |
| a. Dynamic. | b. Static. | c. Compile. | d. Heap. |
|  | In the context of supporting iteration on a stack, what does the term "iterator" refer to? | | | | Answer:  c |
| a. An algorithm for sorting elements. | b. A method to remove elements from the stack. | c. An object used to traverse the elements of the stack. | d. A data structure to store elements temporarily. |
|  | If the input to selection sort is  A = [7, 23, 25, 13, 2, 12, 3, 16, 43] then after two swaps A =... | | | | Answer:  b |
| a. [2, 13, 25, 3, 7, 12, 16, 43] | b. [2, 3, 25, 13, 7, 12, 23, 16, 43] | c. [2, 23, 25, 13, 7, 12, 16, 43] | d. [7, 23, 25, 13, 2, 12, 3, 16, 43] |
|  | Which of the following best describes the time complexity of the selection sort algorithm for sorting an array of size n? | | | | Answer:  c |
| a. | b. | c. | d. |
|  | If the given array is already sorted in ascending order, how many swaps will the selection sort algorithm perform? | | | | Answer:  c |
| a. | b. | c. | d. |
|  | The number of comparisons in the selection sort algorithm is... | | | | Answer:  c |
| a. | b. | c. | d. |
|  | What is the worst-case time complexity of the insertion sort algorithm? | | | | Answer:  c |
| a. | b. | c. | d. |
|  | In insertion sort, the sorted subarray... | | | | Answer:  a |
| a. Starts with the smallest element. | b. Starts with the largest element. | c. Starts with the median element. | d. Starts with a single element. |
|  | In the insertion sort algorithm, how many comparisons are made to sort an already sorted array of size n? | | | | Answer:  a |
| a. | b. | c. | d. |
|  | Shell sort is an extension of which sorting algorithm? | | | | Answer:  b |
| a. Merge sort | b. Insertion sort | c. Quick sort | d. Selection sort |
|  | Which of the following functions grows the fastest as n increases to infinity? | | | | Answer:  d |
| a. | b. | c. | d. |
|  | If an algorithm has a time complexity of , what does this imply? | | | | Answer:  b |
| a. The algorithm always takes exactly operations to complete. | b. The algorithm takes at most operations to complete. | c. The algorithm takes at least operations to complete. | d. The algorithm takes exactly operations to complete. |

**Question 2**

Consider the following array:  
[7, 23, 25, 13, 2, 12, 3]  
Show a trace of execution for insertion sort.

The trace should include the initial state of the array, followed by the array's state after each pass is made.

***Answer:***

initial: [7, 23, 25, 13, 2, 12, 3]

i = 1: [7, 23, 25, 13, 2, 12, 3] // 0

i = 2: [7, 23, 25, 13, 2, 12, 3] // 0

i = 3: [7, 13, 23, 25, 2, 12, 3] // 2

i = 4: [2, 7, 13, 23, 25, 12, 3] // 4

i = 5: [2, 7, 12, 13, 23, 25, 3] // 3

i = 6: [2, 3, 7, 12, 13, 23, 25] // 5

total swaps: 14

**Question 3**

Consider the following array:  
[7, 23, 25, 13, 2, 12, 3]  
Show a trace of execution for selection sort.

The trace should include the initial state of the array, followed by the array's state after each pass is made.

***Answer:***

initial: [7, 23, 25, 13, 2, 12, 3]

i = 0: [2, 23, 25, 13, 7, 12, 3]

i = 1: [2, 3, 25, 13, 7, 12, 23]

i = 2: [2, 3, 7, 13, 25, 12, 23]

i = 3: [2, 3, 7, 12, 25, 13, 23]

i = 4: [2, 3, 7, 12, 13, 25, 23]

i = 5: [2, 3, 7, 12, 13, 23, 25]

i = 6: [2, 3, 7, 12, 13, 23, 25]