**CHAPTER 10**

**Question 1**

What will a program do if an exception is not caught?

Correct Answer:

terminate abnormally

**Question 2**

What section of a try-catch-final region is executed no matter what exceptions are thrown?

Correct Answer:

finally block

**Question 3**

The Error and the Exception classes are subclasses of which class?

Correct Answer:

Throwable

**Question 4**

What statement is used to begin an exception?

Correct Answer:

Throw

**Question 5**

0 out of 1 points

\_\_\_\_\_\_\_\_\_\_ can be used to find the exact line where an exception was thrown during execution.

Correct Answer:

A call-stack trace

**Question 6**

What is used to indicate a block of statements that may throw an exception?

Correct Answer:

try block

**Question 7**

What code block is used to specify how different exceptions should be handled?

Correct Answer:

catch block

**Question 8**

Which of the following exception is unchecked?

Correct Answer:

RuntimeException

**Question 9**

Which of the following exception types must always be caught, unless they are contained in methods that throw them in the method header?

Correct Answer:

Checked

**Question 10**

Which of the following file streams should be explicitly closed to ensure that written data is properly saved?

Correct Answer:

output

**CHAPTER 11**

**Question 1**

Asymptotic complexity describes the general behavior of an algorithm as:

Correct Answer:

n increases.

**Question 2**

Consider the following code:

int i = 0;

while(i < n)

{

System.out.println("Hello world!");

i++;

}

Which of following options gives the Big-Oh of the above loop?

Correct Answer:

O(n)

**Question 3**

Consider the following code:

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

System.out.println("Nested loops!");

Which of following options gives the Big-Oh of the above loop?

Correct Answer:

O(n^2)

**Question 4**

Consider the problem of preparing an order of drinks one at a time. Suppose making each drink takes 2 minutes and putting it in a cup takes an additional 1 minute, and we want to find out how long it will take to complete an order of 5 drinks. If the size of a problem is represented by n, what is the value of n in the given problem?

Correct Answer:

5

**Question 5**

If the following growth function is represented with Big-Oh notation, which of the given Big-Oh terms is correct for the following growth function?

f(n) = 15n3 + 30n2 + 5n + 60

Correct Answer:

O(n^3)

**Question 6**

In the following growth function, what is the dominant term?

f(n) = 15n3 + 30n2 + 5n + 60

Correct Answer:

15n^3

**Question 7**

The analysis of nested loops must take into account both the inner and outer loops.

Correct Answer:

True

**Question 8**

What is the efficiency of an algorithm usually expressed in terms of?

Correct Answer:

use of CPU time by the algorithm

**Question 9**

The order of an algorithm is found by?

Correct Answer:

eliminating constants and all but the dominant term in the algorithm's growth function.

**Question 10**

The Big-Oh order of an algorithm provides \_\_\_\_\_\_\_\_\_ bound to the algorithm’s growth function.

Correct Answer:

an upper

**WEEK 4 SHORT ANSWER**

**Question 1**

Read about the following exception classes in the online Java API documentation and describe their purpose:

ArithmeticException

Thrown when an exceptional arithmetic condition has occurred. For example, an integer "divide by zero" throws an instance of this class. ArithmeticException objects may be constructed by the virtual machine as if suppression were disabled and/or the stack trace was not writable.

NullPointerException

Thrown when an application attempts to use null in a case where an object is required. These include:

Calling the instance method of a null object.

Accessing or modifying the field of a null object.

Taking the length of null as if it were an array.

Accessing or modifying the slots of null as if it were an array.

Throwing null as if it were a Throwable value.

Applications should throw instances of this class to indicate other illegal uses of the null object. NullPointerException objects may be constructed by the virtual machine as if suppression were disabled and/or the stack trace was not writable.

NumberFormatException

Thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format.

IndexOutOfBoundsException

Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range.

Applications can subclass this class to indicate similar exceptions.

**Question 2**

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

5n3 + 3n2 - 3

2n2 + 5n + 17

1000n + 30n2

10n2 log n

Correct Answer:

O(n^3)

O(n^2)

O(n^2)

O(n^2logn)

**Question 3**

Arrange the growth functions of the previous question in ascending order of efficiency for the fixed problem sizes n=10 and again for n = 1000000.

Correct Answer:

Let:

f1(x) = 5n^3+3n^2-3

f2(x) = 2n^2+5n+17

f3(x) = 1000n+30n^2

f4(x) = 10n^2log10(n)

n=10:

f2(10) = 267

f1(10) = 5297

f3(10) = 13000

f4(10) ~= 3321.928

n=1000000:

f2(1000000) = 2.000005\*10^12

f3(1000000) = 3.0001\*10^13

f4(1000000) ~= 1.993\*10^14

f1(1000000) = 5.000003\*10^18

**Question 4**

Determine the growth function and order of the following code fragment:

for (int i=0; i < n; i++)

for (int j=1; j < n; j=j\*2)

System.out.println(i, j);

Correct Answer:

The tight upper bound would be O(n\*log2(n)). Note that since O is an upper bound, any function that bounds n\*log2(n) will also suffice, e.g.: n^2, n^3, 2^n, etc. Also note that since we are interested in asymptotics, the type of log being taken does not matter. (Log conversion between bases is a constant factor.)

**CHAPTER 12**

**Question 1**

In a stack data structure, the last element to be pushed on the stack is the \_\_\_\_\_\_\_\_ one to be removed.

Correct Answer:

first

Response Feedback:

In a stack, the last element to be put on a stack is the first one to be removed.

**Question 2**

In the text's ArrayStack implementation, what exception is thrown if the pop method is called on an empty stack?

Correct Answer:

EmptyCollectionException

Response Feedback:

If a pop method is called on an empty stack, the EmptyCollectionException is thrown.

**Question 3**

Which of the following is not a valid postfix expression?

Correct Answer:

5 - 3

Response Feedback:

4 + 5 is not a postfix expression, but it is an infix expression.

**Question 4**

Which of the following is not the name of a typical stack operation?

Correct Answer:

dequeue

Response Feedback:

dequeue is an operation on queue and not on a stack.

**Question 5**

Abstractions hide an object's internal data, and how it implements operations. These ideas are also part of?

Correct Answer:

Encapsulation.

Response Feedback:

From the concept of Abstraction in OO programming. Abstraction hide (encapsulate) the object's data and the implementation of the operations.

**Question 6**

All objects are abstractions in that they provide well-defined operations. These operations may be represented in?

Correct Answer:

an interface

Response Feedback:

From the concept of Abstraction in OO programming. All objects are abstractions in that they provide well-defined operations (the interface)

**Question 7**

Stacks are a \_\_\_\_\_\_\_\_ type of Collection.

Correct Answer:

Linear

**Question 8**

Which of the following images correctly represents a conceptual view of a stack?

Correct Answer:

(a) Stack conceptual view 1

Response Feedback:

The image (a) Stack conceptual view a represents the correct conceptual view of a stack. Stack has only top from where elements can be added (push) or removed (pop)

**Question 9**

If a stack was used to record method as they are called in a program, then a push would representing calling a method, and a pop would represent a method completing.

Correct Answer:

True

**Question 10**

Which one of the following is an example of a complete javadoc style comment?

Correct Answer:

Correct

/\*\*

\* These are javadoc comments

\*/

**CHAPTER 13**

Question 1

Programming language support for references is important for implementing linked lists.

Correct Answer:

True

**Question 2**

A linked list where each node contains both forward and backward references would be called a \_\_\_\_\_\_\_\_.

Correct Answer:

doubly-linked list

**Question 3**

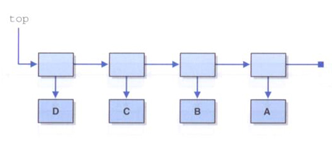
When a class, called X, contains a variable of type X, we say that the class is \_\_\_\_\_\_\_\_.

Correct Answer:

self-referential

**Question 4**

Consider the following figure that shows a linked stack:



Which of the following sequence of code will result the stack as shown in figure above? Assume that a stack called myStack has already been initialized.

Correct Answer:

myStack.push(“A”);

myStack.push(“B”);

myStack.push(“C”);

myStack.push(“D”);

**Question 5**

If you needed to store 25 names of students, and wanted to access them quickly, the best data structure would most likely be:

Correct Answer:

An array.

**Question 6**

A linked list implementation of a stack needs to store references to both the first and last element of the list.

Correct Answer:

False

**Question 7**

In the singly linked list described in the text, how many other Nodes can potentially be accessed (directly or indirectly) if you start from a single Node?

Correct Answer:

0 or more

**Question 8**

If you were to compare the amount of memory needed to store an array of 100 integers and a linked list containing 100 integers, then:

Correct Answer:

The list would require more memory than the array.

**Question 9**

If you were optimizing for performance, and wanted to access the 500th element of an array or list, then:

Correct Answer:

An array would allow faster access than the list.

**Question 10**

If you were optimizing for performance and wanted to support potentially adding many new elements to a data structure, then:

Correct Answer:

The list would allow faster addition of elements than the array.

**WEEK 5 SHORT ANSWERS**

**Question 1**

Compare and contrast: data types, abstract data types and data structures. Include how they are related.

Correct Answer:

A data type is meta-information about a value that defines how that piece of data behaves. Typically, data types define a whole class of values (e.g., integers, characters, etc.) that operate with the same behavior. An abstract data type is similar to the idea of a collection interface - it is a definition of a type in terms of its behavior. Abstract data types do not define how such behavior is implemented, they are instead a layer of abstraction between a programmer requiring certain behavior from data, and how that behavior is implemented. Data structures are (typically) concrete implementations of abstract data types.

**Question 2**

Trace an initially empty Stack (called S) through the following operations:

Stack S = new Stack();

S.push(new Integer(4));

S.push(new Integer(3));

//a) What is the content of the stack at this point?

Integer Y = S.pop();

S.push(new Integer(7));

S.push(new Integer(2));

S.push(new Integer(5));

S.push(new Integer(9));

//b) What is the content of the stack at this point?

Integer Y = S.pop();

S.push(new Integer(3));

S.push(new Integer(9));

//c) What is the content of the stack at this point?

Correct Answer:

(top) 3 4

(top) 9 5 2 7 4

(top) 9 3 5 2 7 4

**Question 3**

Assume you have a stack called S with the following contents, and trace the following operations:

4 (top)

8

2

3

1

System.out.println(S.peek()); //a) What is printed?

S.pop();

System.out.println(S.peek()); //b) What is printed?

System.out.println(S.pop()); //c) What is printed?

System.out.println(S.peek()); //d) What is printed?

//e) What are final the contents of the stack?

Correct Answer:

4

8

8

2

(top) 2 3 1

**Question 4**

Explain how the undo operation in a word processor can be supported by the use of a stack. Give specific examples and comment on the contents of the stack after various actions are taken.

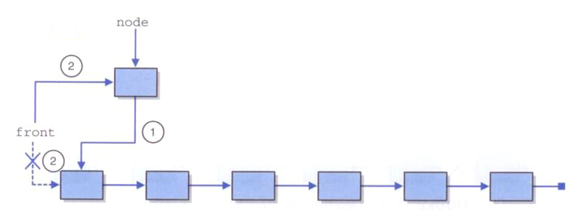
Correct Answer:

The stack will store strings, and text will be represented as a traversal from the front to the top. As words (or letters; depends on required granularity) are entered, they are pushed onto a stack. When an undo operation is required, an item can be popped from the stack. This removes the most recent string and leaves the rest.

**Question 5**

The figure below shows a two step process to insert a node at the front of a linked list. Assume that front is a reference to the first node of the list and node is a variable containing the new node. Step 1: Assign front as node's next node (node.setNext(front);). Step 2: Assign the new node to front (front = node;) .

Explain what will happen if the steps are reversed.



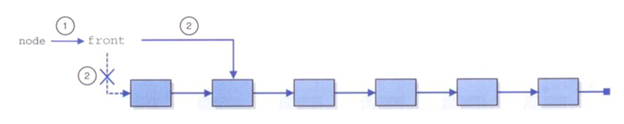
Correct Answer:

The existing list would be replaced with node, which will contain a self-loop.

**Question 6**

The figure below shows a two step process for removing a node from a list while keeping a reference to it. Assume that front is a reference to the first node of the list and node is a variable that will record the removed node. Step 1: Assign the front variable to the node variable (node = front;). Step 2: Assign the second node in the list to the front variable (front = front.getNext();).

Explain what will happen if the steps are reversed.



Correct Answer:

The list would lose its first node correctly, however, the node variable would be a reference to the second node in the original list (the first node in the reduced list).

**Question 7**

Needs Grading

Consider the following code for creating and inserting a element into a list. Assume that front is a variable that represents a reference to the first node of a list.

public void insert\_number(T element) {

LinearNode<T> temp = new LinearNode<T>(element);

//insert

temp = front;

front.setNext(temp); }

Answer the following questions:

a) Does this function correctly insert the node into the linked list? Explain. Hint: Try drawing a picture of the list.

b) Give the correct code for adding node at the head of the linked list.

Correct Answer:

No, it will result in the current list being reduced to the first element with a self-loop. The new node will be lost. (The code will also crash if the existing list is empty).

public void insert\_number(T element) {

LinearNode<T> temp = new LinearNode<T>(element);

//insert (at head solution)

temp.setNext(front);

front = temp;}

**Question 8**

Consider the following scenario: In a social media news feed, users are required to subscribe to the news feed before receiving any news or updates. Users can only subscribe to each news feed only once up to five (5) different feeds.

Given your knowledge of linked structures and arrays, determine which of the two complex types you would use to implement the solution (1) for tracking the number of subscribers within a media company and (2) for tracking the number of news feed a subscriber can subscribe to. Justify your answers. (e.g., identify the advantages and disadvantage for inserting, deleting, and searching through different complex types.)

Correct Answer:

For tracking the subscribers, a listed link is a good choice. The key is to allow fast insertion and deletion. We can expect that there may be a high number of subscribers. The issue is that the total number will fluctuate as service subscribers come and go. If we used an array, we would have to spend a lot of time resizing it and copying the memory allocated to entries. Using a linked list means we can adjust the subscribers without having to reallocate all the other subscribers.

For tracking the news feeds, an array is a good choice. In general, arrays are simply fast. They allocate at one time and the only computation need to access it is an index pointer addition. It is also lighter weight since the data structure it stores does not need to have pointers for a 'next' linked list node. The limitation of arrays is that we cannot increase its size without building an entirely new array. However, in this case we know that the user can only subscribe to five (5) feeds – creating an array of that size is sufficient to hold the feed without any size issues.

**CHAPTER 14**

**Question 1**

In an array-based implementation of a queue that always stores the front of the queue at index 0 in the array, what would be the time complexity (Big-Oh) of the dequeue operation. (Assume the array isn't circular.)

Correct Answer:

O(n)

Response Feedback:

It requires a linear amount of time to shift all elements of the queue down an index after a remove is applied. So the operation is O(n).

**Question 2**

If a queue is implemented using a circular array approach, the elements must all be shifted when the dequeue operation is called.

Correct Answer:

False

Response Feedback:

A circular array-based implementation of a queue eliminates the need to shift elements, so all queue operations can be achieved in constant time.

**Question 3**

1 out of 1 points

In an ideal implementations of a stack and a queue, complexity of all operations is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ .

Correct Answer:

O(1)

Response Feedback:

In good implementations of stacks and queues, all operations require a constant amount of time, that is O(1).

**Question 4**

Is it possible for the front and rear references in a circular array implementation to be equal?

Correct Answer:

Yes, it can happen under two circumstances: when the queue is empty, and when the queue is full.

**Question 5**

The only way possible to implement a queue is using an array-based structure.

Correct Answer:

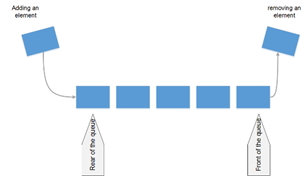
False

Response Feedback:

A queue can be implemented using a linked structure or an array-based structure.

**Question 6**

Which of the following image correctly represents a conceptual view of a queue?



Response Feedback:

The image (a) Queue conceptual view a represents the correct conceptual view of a Queue. Stack has a Rear from where elements can be added (enqueue) and a Front end from where an element can be removed (dequeue)

**Question 8**

Which of the following methods removes an element from a queue?

Correct Answer:

dequeue

Response Feedback:

The dequeue method removes an element from a queue.

**Question 9**

Which of the following data structures would be most appropriate for simulating a waiting line?

Correct Answer:

Queue

Response Feedback:

Simulations are often implemented using queues to represent waiting lines.

**Question 10**

1 out of 1 points

Which type of data structure a queue is?

Correct Answer:

FIFO

Response Feedback:

A queue is a FIFO (first in, first out) data structure, meaning that the first element that is put into the queue is the first element to be removed from the queue.

**CHAPTER 18**

**Question 1**

In a binary search, what property of the search pool is assumed?

Correct Answer:

the search pool is ordered.

Response Feedback:

In a binary search, it is assumed that the search pool is ordered.

**Question 2**

What is the process of finding a designated target element within a group of items?

Correct Answer:

Searching

Response Feedback:

Searching is the process of finding a designated target element in a search pool.

**Question 3**

What process arranges a group of items into a defined order?

Correct Answer:

Sorting

**Question 4**

Which algorithm sorts a list of values by repetitively inserting a particular value into a subset of the list that has already been sorted?

Correct Answer:

insertion sort

Response Feedback:

Insertion sort works in the way described in the question.

**Question 5**

Which algorithm sorts values by repeatedly comparing neighboring elements in the list and swapping their position if they are not in order relative to each other?

Correct Answer:

bubble sort

Response Feedback:

Bubble sort works in the way described in the question.

**Question 6**

Which type of search looks through the search pool one element at a time?

Correct Answer:

linear

Response Feedback:

In a linear search, the algorithm iterates through the pool one at a time in a linear manner.

**Question 7**

Which search is more efficient than a linear search?

Correct Answer:

binary

Response Feedback:

A binary search is more efficient than a linear search, but it assumes that the search pool is ordered.

**Question 8**

Which of the following algorithms has a time complexity of O(log n)?

Correct Answer:

binary search

Response Feedback:

Binary search has a time complexity of O(log n).

**Question 9**

Insertion sort is normally considered an O(n^2) algorithm. If the algorithm was run on an array that was already sorted, would the algorithm really need n^2 steps to complete?

Correct Answer:

No. If the array was sorted, then no elements would need to be moved.

**Question 10**

Merge sort is normally considered an O(n log n) algorithm. If the algorithm was run on an array that was already sorted, would the algorithm really need n log n steps to complete?

Correct Answer:

Yes, it will still require on the order of n log n operations. The order of the array does not have a significant impact on the number of the recursive calls or the execution of the while loops in merge.

**WEEK 6 SHORT ANSWER**

**Question 1**

Needs Grading

Trace a Queue (called Q) through the following operations:

Queue Q = new Queue();

Q.enqueue(new Integer(4));

Q.enqueue(new Integer(3));

//a) What is the content of the queue at this point?

Integer Y = Q.dequeue();

Q.enqueue(new Integer(7));

Q.enqueue(new Integer(2));

Q.enqueue(new Integer(5));

Q.enqueue(new Integer(9));

//b) What is the content of the queue at this point?

Integer Y = Q.dequeue();

Q.enqueue(new Integer(3));

Q.enqueue(new Integer(9));

//c) What is the content of the queue at this point?

Correct Answer:

3, 4 (front)

9, 5, 2, 7, 3 (front)

9, 3, 9, 5, 2, 7 (front)

**Question 2**

Needs Grading

Assume you have a queue called Q with the following contents, and trace the following operations:

(front) (back)

4 8 2 3 1

System.out.println(Q.first()); //a) What is printed?

Integer Y = Q.dequeue();

System.out.println(Q.first()); //b) What is printed?

Q.enqueue(new Integer (5));

System.out.println(Q.dequeue()); //c) What is printed?

System.out.println(Q.first()); //d) What is printed?

//e) What are final the contents of the queue? (Indicate front and back.)

Correct Answer:

4

8

8

2

5, 1, 3, 2 (front)

**Question 3**

Compare and constrast the enqueue method of the LinkedQueue class to the push method of the LinkedStack class from Chapter 13.

Correct Answer:

Implementation-wise, they are virtually identical. Conceptually they are both adding an element to the end of a linear data structure. However, stacks and queues have different functionality (dequeue vs pop) that is defined with respective to enqueue/push (i.e. if operations happen on the same or different end of the linear data structure).

**Question 4**

Explain why the array implementation of a stack does not require elements to be shifted but the (non-circular) array implementation of a queue does.

Correct Answer:

In a stack elements are removed and added at the same position - neither operation has an impact on the other elements. In a queue, elements can be add and removed from different places. The plain array implementation of a queue, enables this by fixing the position of the first element of the queue at the start of the array. This allows new elements to be add at the end easily but requires that elements be shifted when a dequeue occurs.

**Question 5**

Compare the linearSearch and binarySearch algorithms by searching for the numbers 8, 33, 84 in the array 4, 8, 12, 23, 54, 84, 89, 110. For both algorithms, list (in order) the elements of the array that the algorithm will check against target before finding the correct result. If an element cannot be found, note where the algorithm will stop.

Correct Answer:

Linear:

8 - 4, 8

33 - 4, 8, 12, 23, 54, 84, 89, 110 (stop; not found)

84 - 4, 8, 12, 23, 54, 84

Binary (depends on rounding):

8 - 23, 8

33 - 23, 84, 54 (stop; not found)

84 - 23, 84

**Question 6**

Consider the following array: 97, 8, 7, 45, 133, 325, 9, 1, 563.

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 swap is made.

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

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

Merge sort. Illustrate how the array is broken down, and then merged into an ordered state. The trace should follow the format shown in Figures 18.5 and 18.6.

Correct Answer:

Selection Sort:

0) 97, 8, 7, 45, 133, 325, 9, 1, 563 (~~final initial~~)

1) 1| 8, 7, 45, 133, 325, 9, 97, 563

2) 1, 7| 8, 45, 133, 325, 9, 97, 563

3) 1, 7, 8| 45, 133, 325, 9, 97, 563

4) 1, 7, 8, 9| 133, 325, 45, 97, 563

5) 1, 7, 8, 9, 45| 325, 133, 97, 563

6) 1, 7, 8, 9, 45, 97| 133, 325, 563

7) 1, 7, 8, 9, 45, 97, 133| 325, 563

8) 1, 7, 8, 9, 45, 97, 133, 325| 563

9) 1, 7, 8, 9, 45, 97, 133, 325, 563|

Insertion Sort:

0) 97, 8, 7, 45, 133, 325, 9, 1, 563 (inital)

1) 97| 8, 7, 45, 133, 325, 9, 1, 563

2) 8, 97| 7, 45, 133, 325, 9, 1, 563

3) 7, 8, 97| 45, 133, 325, 9, 1, 563

4) 7, 8, 45, 97| 133, 325, 9, 1, 563

5) 7, 8, 45, 97, 133| 325, 9, 1, 563

6) 7, 8, 45, 97, 133, 325| 9, 1, 563

7) 7, 8, 9, 45, 97, 133, 325| 1, 563

8) 1, 7, 8, 9, 45, 97, 133, 325| 563

9) 1, 7, 8, 9, 45, 97, 133, 325, 563|

Bubble:

0) 97, 8, 7, 45, 133, 325, 9, 1, 563 (inital)

1) 8, 7, 45, 97, 133, 9, 1, 325, 563

2) 7, 8, 45, 97, 9, 1, 133, 325, 563

3) 7, 8, 45, 9, 1, 97, 133, 325, 563

4) 7, 8, 9, 1, 45, 97, 133, 325, 563

5) 7, 8, 1, 9, 45, 97, 133, 325, 563

6) 7, 1, 8, 9, 45, 97, 133, 325, 563

7) 1, 7, 8, 9, 45, 97, 133, 325, 563

8) 1, 7, 8, 9, 45, 97, 133, 325, 563

Merge Sort:

0) [97, 8, 7, 45, 133, 325, 9, 1, 563] (inital)

1) [[97, 8, 7, 45, 133][325, 9, 1, 563]]

2) [[[97, 8, 7][45, 133]][[325,9][1,563]]]

3) [[[[97, 8][7]][[45][133]]][[[325][9]][[1][563]]]]

4) [[[[[97][8]][7]][[45][133]]][[[325][9]][[1][563]]]]

(start merging)

5) [[[[8, 97][7]][45, 133]][[9, 325][1, 563]]]

6) [[[7, 8, 97][45, 133]][[9, 325][1, 563]]]

7) [[[7, 8, 97][45, 133]][1, 9, 325, 563]]

8) [[7, 8, 45, 97, 133][1, 9, 325, 563]]

9) [1, 7, 8, 9, 45, 97, 133, 325, 563]