**Heap’s Algorithm for generating permutations**

A permutation, also called an “arrangement number” or “order,” is a rearrangement of the elements of an ordered list S into a one-to-one correspondence with S itself. A string of length n has n! permutation.

Source: Mathword(<http://mathworld.wolfram.com/Permutation.html>)

Below are the permutations of string ABC:

ABC ACB BAC BCA CBA CAB

ALGORITHM:

Method: permute - word, start, end - void

IF: start is equal to end

THIS IS A NEW PERMUTATION

ELSE:

FOR start-->end

set word to: RUN: swap for word, start, i

RUN permute again for next start index

set word to: RUN: swap for word, start, i // to set the word back to its original value if needed

Method: swap - word, i, j - returns String

new char array equals word to Char Array

new char temp equals i

char i equals char j

char j equals char temp

return string value of char array

**BINARY TO CHAR**

(char)Integer.parseInt(s.substring(i, i+8), 2)

**CHAR TO ASCII**

int asciiValue = (int)char

**ASCII TO CHAR**

char letter = (char)ascii

**Conversions**

String.valueOf(); // TO CONVERT ANY VALUE TO A STRING (ARRAYS INCLUDED)

STRING.toCharArray(); // CONVERT ANY STRING TO A CHARACTER AN ARRAY

**What is contains() method in Java?**

The contains() method is Java method to check if String contains another substring or not. It returns boolean value so it can use directly inside if statements.

Syntax of String "Contain" method: public boolean String.contains(CharSequence s)

**ARRAYLIST INFO**

**Constructors in Java ArrayList:**

ArrayList(): This constructor is used to build an empty array list

ArrayList(Collection c): This constructor is used to build an arraylist initialized with the elements from collection c

ArrayList(int capacity): This constructor is used to build an arraylist with initial capacity being specified

**Methods in Java ArrayList:**

forEach​(Consumer action): Performs the given action for each element of the Iterable until all elements have been processed or the action throws an exception.

retainAll​(Collection c): Retains only the elements in this list that are contained in the specified collection.

removeIf​(Predicate filter): Removes all of the elements of this collection that satisfy the given predicate.

contains​(Object o): Returns true if this list contains the specified element.

remove​(int index): Removes the element at the specified position in this list.

remove​(Object o): Removes the first occurrence of the specified element from this list, if it is present.

get​(int index): Returns the element at the specified position in this list.

subList​(int fromIndex, int toIndex): Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

spliterator​(): Creates a late-binding and fail-fast Spliterator over the elements in this list.

set​(int index, E element): Replaces the element at the specified position in this list with the specified element.

size​(): Returns the number of elements in this list.

removeAll​(Collection c): Removes from this list all of its elements that are contained in the specified collection.

ensureCapacity​(int minCapacity): Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

listIterator​(): Returns a list iterator over the elements in this list (in proper sequence).

listIterator​(int index): Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.

isEmpty​(): Returns true if this list contains no elements.

removeRange​(int fromIndex, int toIndex): Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive.

void clear(): This method is used to remove all the elements from any list.

void add(int index, Object element): This method is used to insert a specific element at a specific position index in a list.

void trimToSize(): This method is used to trim the capacity of the instance of the ArrayList to the list’s current size.

int indexOf(Object O): The index the first occurrence of a specific element is either returned, or -1 in case the element is not in the list.

int lastIndexOf(Object O): The index the last occurrence of a specific element is either returned, or -1 in case the element is not in the list.

Object clone(): This method is used to return a shallow copy of an ArrayList.

Object[] toArray(): This method is used to return an array containing all of the elements in the list in correct order.

Object[] toArray(Object[] O): It is also used to return an array containing all of the elements in this list in the correct order same as the previous method.

boolean addAll(Collection C): This method is used to append all the elements from a specific collection to the end of the mentioned list, in such a order that the values are returned by the specified collection’s iterator.

boolean add(Object o): This method is used to append a specific element to the end of a list.

boolean addAll(int index, Collection C): Used to insert all of the elements starting at the specified position from a specific collection into the mentioned list.

Class Name implements Comparable<Name>

declare num, string, double, etc.

compareTo object of type Type Name return int

return Type.compare() for int, double etc.

return use Name.compareTo(other string) for strings

Will return 1 for first value greater, 0 for same value, and -1 for first value is less

So could do:

int equals comparison one

If int equals 0, then int equals comparison two

return int

This would allow sorts based on an order of multiple comparisons

Call Arrays.sort(Object of type name) to sort