```
Lab 8: Implementing Sort Algos
Due in your lab during the week of March 26
Write all of your functions into one file called "sortlib.py"
and submit it with an arg of 8.

See below for the API specs for the functions you must implement.
For all of the below, you are to complete the "..." bodies of
the indicated algo, where the sort is to produce a list in
ascending order. Some of the functions are used in others,
you will see the relationship once you completely read the spec.
[Quick Sort and Partition are NOT required for this week's lab]

You may include arbitrary helper functions in the file, provided
they have the prefix helper_

* API: Application Programming Interface
An API provides the interfaces for functions that you will be
exposing for a user to use to write an application.
For example, below, I am asking you to write a set of sort functions.
I will write a test application that will call your sort functions
(in industry, in addition to someone writing a test app, someone else
will write an application that utilizes your functions for some
purpose).
Since there are untold numbers of permutations that a implementer
(i.e., you) may choose, we require a specification that clarifies
the input and output characteristics. This is called an API.

Sorting is relatively straightforward so what you need to know is
* how I will be providing the data to you (a list, below called u)
* what I expect as output
   * True or False to be returned
   * the sorted list returned via the arg list
     (that is, you will be changing "u" inside your algo; this is
     called doing a sort "in-place")
Recall in CSC180 I said that you should never change your arguments
in Python. That is correct, you shouldn't. However, with sorting there
is often a major benefit to be gained by sorting inplace (if you have
a large input list, you will not need to copy it to a new output list)
and in fact many algorithms are designed to be done inplace.
Hence, with this lab, I'm asking you to also do the work inplace.
Since we have specified this via the API (i.e., the spec), this is not
a problem. It will always be a bad idea to do something in place if you
do not explicitly specify that it will be done in place.


def selection_sort(u):
   ...
   return True

def heapify(u):
   ...
   return True

def reheapify(u,end):
   ...
   return True

def heap_sort(u):
   ...
   heapify(u)
   ...
   ... some loop ...
      ...
      reheapify(u,end)
      ...
   ...
   return True

def merge_sort(u):
   ...
   return True

def quick_sort(u,ini,fin):
   ...
   pIndex = partition(u,ini,fin)
   ...
   return True

def partition(u,ini,fin):
   ...
   return pIndex

Example test code:

v1=[10,9,8,7,6,5,4,3,2,1,0]
v2=[100,1,1000,9,8,7,2,2000,10]
v3=[100,10,1000,9,8,7,2,6,5,2,3,1]

for i in [ v1,v2,v3 ]:
   x=list(i)
   selection_sort(x)
   print x

   x=list(i)
   heapify(x)
   print x

   x=list(i)
   heap_sort(x)
   print x

   x=list(i)
   merge_sort(x)
   print x

   x=list(i)
   quick_sort(x,0,len(x)-1)
   print x
```