CS 241 DATA STRUCTURES FALL 2021 MIDTERM II
Instructor Calvin Deutschbein

Roster Name	
Sign here to affirm the Honor Code	

This exam will be timed to take 60 Minutes.

It will be scored out of 200 Points.

It will make up 20% of Final Grade.

SECTION I: PYTHON **40 Points**

Part 1: Multiple Choice: *4 Questions @ 5 Points each =* *20 Points*

Which of the following creates an empty **Hash Table** using “hash.py” from HW4?

A. >>> h = HashTable()
B. >>> h = HashTable.__init__()
C. >>> h = []
D. >>> h = [None] * size

Which of the following creates an empty **Linked List** using SortedNatList.py from HW1?

A. >>> ll = SortedNatList()
B. >>> ll = SortedNatList.__init__()
C. >>> ll = []
D. >>> ll = [None] * size

What is the **return type** of Python sort() when used on a list, such as somelist.sort()?

A. None
B. List
C. Boolean
D. Integer

What is the **return type** of Python sorted() return when applied to list, such as sorted(somelist)?

A. None
B. List
C. Boolean
D. Integer

Part 2: Short Response: 2 Questions @ 10 Points each = 20 Points

Hash tables are one way to store unsorted data. Describe the usefulness of hashing functions, such as the built-in Python hash(), for storing data.

Python contains built-in methods to sort data, such as sorted() and sort(). However, these methods rely on using other built-in methods, such as the greater than and less than operators.

Suppose you wanted to sort a list of strings by length, rather than alphabetically. Could you use sorted() or sort()? Why or why not?

SECTION II: ORGANIZING DATA **40 Points**

Part 3: True/False: *4 Questions @ 5 Points each =* *20 Points*

Specify whether the code snippets return boolean values True or False.

```
>>> len([1]) <= 1
```

A. True
B. False

```
>>> type(set()) == type([])
```

A. True
B. False

```
>>> if True:
. . .       True
. . .   else:
. . .       False
```

A. True
B. False

```
>>> if False:
. . .       True
. . .   else:
. . .       False
```

A. True
B. False

Part 4: Free Response: *20 Points*

Bubble sort is an algorithm to sort lists by

- Stepping through the list
- Swapping adjacent elements that are not in order
- Repeating this process until the list is sorted

```
def bsort(l):
     def swap(l,i,j):
          if l[i] > l[j]:
               l[i], l[j] = l[j], l[i]
               return True
          return False
     swapping = True
     while swapping:
          swapping = False
          for i in range(len(l) - 1):
               swapping = swapping or swap(l,i,i+1)
     return
```

How would you expect the performance of bubble sort to compare to insertion sort, quick sort, heap sort, and merge sort? Consider the idea of “Big O notation” when you answer. Consider best, worst, and average cases.

SECTION III: SORTING DATA **70 Points**

Part 5: Multiple Choice: *4 Questions @ 5 Points each =* *20 Points*

Given a shuffled list of a million integers, what is the fastest way to sort them?

A. Insertion sort
B. Merge sort
C. Heap sort
D. Quick sort

Given a list of a million integers that may be either sorted or shuffled, what is the fastest sort?

A. Insertion sort
B. Merge sort
C. Heap sort
D. Quick sort

Suppose you must add 10 new integers to the list. What is the best way to add these elements to the sorted list so that the resulting list is also sorted?

A. Insertion sort
B. Merge sort
C. Heap sort
D. Quick sort

Suppose you had a computer with a very slow processor (comparisons are expensive) but very fast memory (using structures is inexpensive). What sort may benefit from this arrangement?

A. Insertion sort
B. Merge sort
C. Heap sort
D. Quick sort

Part 6: Written Response: *25 Questions @ 2 Points each =* *50 Points*

Sort	Insert Cost	Contains Cost	Average Sort Cost	Worst Case Sort Cost	Structure
Insert					
					Heap
Merge					
Quick					
					Hashtable

SECTION IV: ALGORITHMS **50 Points**

Part 7: Coding Exercise *50 Points*

Implement a simplified hash table with insert and contains methods. You need not consider removal or rehashing. You may use any built-in Python data structures or anything we wrote in class.