Spring 2013 Name: Mark F.

Data Structures - Test 2

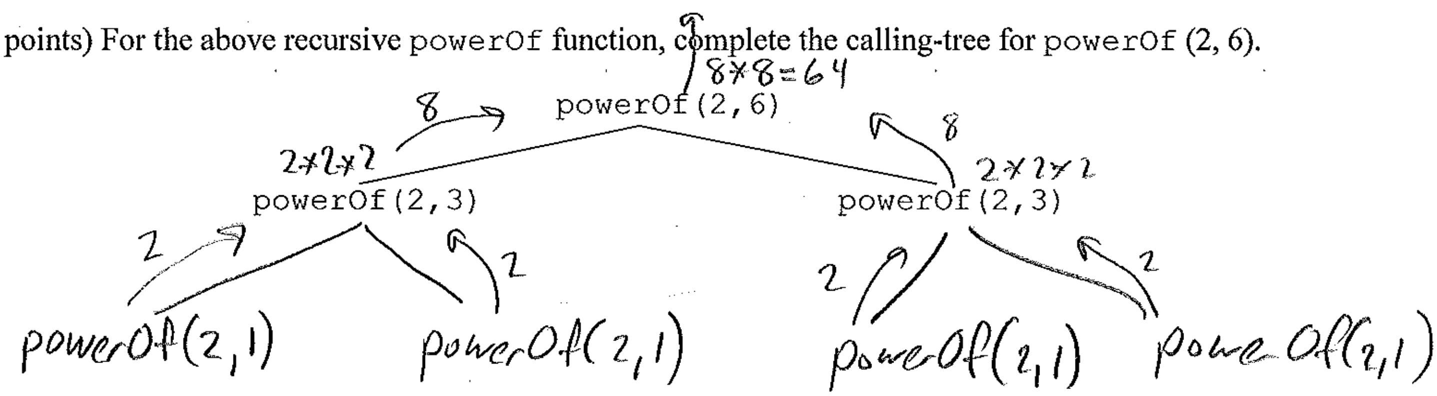
Question 1. Write a recursive Python function to calculate a^n (where n is an integer) based on the formulas:

$a^0 = 1$, for n = 0
$a^1 = a$, for n = 1
$a^n = a^{n/2}a^{n/2}$, for even n > 1
$a^n = a^{(n-1)/2}a^{(n-1)/2}a$, for odd n > 1

a) (12 points) Complete the below `powerOf` **recursive** function

```
def powerOf(a, n):
    if n==0
        return 1
    elif n==1
        return a
    elif (n % 2)==0:
        return powerOf(a, n//2)*powerOf(a, n//2)
    else:
        return powerOf(a, (n-1)//2)*powerOf(a, (n-1)//2)*a
```

b) (8 points) For the above recursive `powerOf` function, complete the calling-tree for `powerOf (2, 6)`.



c) (5 points) Suggest a way to speedup the above `powerOf` function.

- Don't call powerOf twice with same parameters

```
elif
    return powerOf(a, n//2)**2
else
    return (powerOf(a, (n-1)//2)^2)*a
```

- Dynamic programming

25

Question 2. (10 points.) Consider the following insertion sort code which sorts in ascending order.

```
def insertionSort(myList):
    """Rearranges the items in myList so they are in ascending order"""

    for firstUnsortedIndex in range(1,len(myList)):
        itemToInsert = myList[firstUnsortedIndex]

        testIndex = firstUnsortedIndex - 1

        while testIndex >= 0 and myList[testIndex] > itemToInsert:
            myList[testIndex+1] = myList[testIndex]
            testIndex = testIndex - 1

        # Insert the itemToInsert at the correct spot
        myList[testIndex + 1] = itemToInsert
```

a) What initial arrangement of items causes the is the overall worst-case performance of insertion sort?

+2 initially in descending order -- away inserts at index 0   sorted

b) What is the worst-case $O(\,)$ notation insertion sort?

+3 $1+2+3+\dots+(n-3)+(n-2)+(n-1) = n * \frac{(n-1)}{2} \in O(n^2)$

c) What initial arrangement of items causes the is the overall best-case performance of insertion sort?

+2 initially in ascending order already. - while loop never entered.

d) What is the best-case $O(\,)$ notation for insertion sort? $O(n)$

+3

10

Question 3. (25 points) Write a variation of selection sort that:

- sorts in descending order (largest to smallest)
- builds the sorted part on the left-hand side of the list by having each pass of the outer loop do the following:

myL

Sorted Part	Unsorted Part

1) Inner loop that scans the unsorted part to find the index of the largest item in the unsorted part
2) Swap the first item in the unsorted part with the largest item in the unsorted part that was found in (1)

```
def selectionSort(myList):
    for firstUnsortedIndex in range(len(myList)-1):
        maxIndex = firstUnsortedIndex
        for testIndex in range(firstUnsortedIndex+1, len(myList)):
            if myList[testIndex] > myList[maxIndex]:
                maxIndex = testIndex
        temp = myList[firstUnsortedIndex]
        myList[firstUnsortedIndex] = myList[maxIndex]
        myList[maxIndex] = temp
```

Spring 2013 Name: ______________________

Question 4. (20 points) Recall the common rehashing strategies we discussed for open-address hashing:

Strategy	Description
linear probing	Check next spot (counting circularly) for the first available slot, i.e., (home address + (rehash attempt #)) % (hash table size)
quadratic probing	Check the square of the attempt-number away for an available slot, i.e., [home address + ((rehash attempt #)^2 +(rehash attempt #))/2] % (hash table size), where the hash table size is a power of 2. Integer division is used above

a) Insert "Paul Gray" and then "Kevin O'Kane" using Linear (on left) and Quadratic (on right) probing.

Hash Table with Linear Probing

0	Ben Schafer
1	Kevin O'Kane
2	
3	Philip East
4	
5	Mark Fienup
6	John Doe
7	Paul Gray

Hash function

hash(John Doe) = 6
hash(Philip East) = 3
hash(Mark Fienup) = 5
hash(Ben Schafer) = 0
hash(Paul Gray) = 5
hash(Kevin O'Kane) = 6

Hash Table with Quad. Probing

0	Ben Schafer
1	Kevin O'Kane
2	
3	Philip East
4	
5	Mark Fienup
6	John Doe
7	Paul Gray

$5 + \frac{1^2+1}{2} = 6$

$5 + \frac{2^2+2}{2} = 8 \% 8 = 0$

$5 + \frac{3^2+3}{2} = 11 \% 8 = 3$

$5 + \frac{4^2+4}{2} = 15 \% 8 = 7$

b) Explain why the average/expected search time for hashing is O(1).

Hopefully the hash fn. will calculate a unique home addr. for key, so we find it immediately.

$6 + \frac{1^2+1}{2} = 7$

$6 + \frac{2^2+2}{2} = 9 \% 8 = 1$

Question 5. (20 points) Heap sort uses a min-heap to sort a list. (BinHeap methods: `BinHeap()`, `insert(item)`, `delMin()`, `isEmpty()`, `size()`)

Generl idea of Heap sort:

myList: unsorted list with n items

1. Create an empty heap
2. Insert all n list items into heap

heap with n items

3. delMin heap items back to list in sorted order

myList: sorted list with n items

a) What is the overall *O*( ) for heap sort? $O(n \log_2 n)$

b) Explain your O( ) answer for part (a).

Step 1 is O(1)

Step 2 does n calls to insert where each insert is at most $\log_2 n$ loops since thats the height of the heap. Thus, $n * \log_2 n$

Step 3 is similar to Step 2: n calls to delMin where each is $O(\log_2 n)$ $\Rightarrow$ total $n \log_2 n$