University of Illinois at Urbana-Champaign
Department of Computer Science

Final Examination

CS 125 Introduction to Computer Science
Spring 2008
3 Hours

Name: Last, First
NetID: _ _ _ _ _ _ _ _ @ uiuc.edu

PLEASE READ THE FOLLOWING CAREFULLY

• This is a closed book and closed notes exam.
• You are not allowed to use the break, continue, or switch statements on this exam. Furthermore, you cannot use loops in any problem unless we specifically say that you can.
• Unless we say otherwise in the specific problem, you can assume all values entered by the user will be acceptable input for that program.
• When you write code, you may use a shorthand for System.out and TextIO input and output methods provided it is obvious to the graders which method you are using. For example it is acceptable to use Sopln in place of System.out.println and to use Sopt in place of System.out.print Likewise, you can use T.rlnI(), T.rlnC(), and T.rlnD() in place of TextIO.readlnInt(), TextIO.readlnChar(), and TextIO.readlnDouble().
• For full marks correct syntax is required: Ensure all statements include a semicolon and the correct use of upper/lower case, single quotes and double quotes.

Problem	Points	Score	Grader
1	6		
2	14		
3	14		
4	15		
5	12		
6	15		
7	12		
8	12		
Total	100		

1. Guess my number – 6 points.

HAL and Arthur are playing a simple 'guess my number' game:

HAL: "I'm thinking of a integer between 1 and 1000 inclusive"
Arthur: "Is it 500?"
HAL: "No, go lower"
Arthur "Is it 250?
HAL: "No, go higher"
Arthur "Is it 375?
HAL: "Yes"

For the following questions:

- The problem size, 'N' is the number of integers that HAL claims must be considered at the start of the game (i.e. N=1000 in the dialog above).

- Use Big-O notation for your answer e.g. $O(n^2)$. You only need to write the order of growth; you do NOT need to explain how you got your answer.

Q. #1 HAL chooses an unknown integer apparently at random so Arthur uses an intelligent divide and conquer strategy to minimize the number of guesses required. Determine the order of growth for the *worst case*: How does the time required to discover HAL's answer increase as the initial number of possible answers increases?

Q. #2. What is the best case order of growth running time for the same scenario described in Q1? Use big-O notation.

Q. #3 Later HAL changes the rules and responds with just "No, that's not my number" or "Yes". Arthur modifies his algorithm to run optimally given HALs uncooperative responses. Determine the worst case running time of Arthur's algorithm.

2. Recursive Thesaurus – 14 points (8+6).

a) You want to recursively search an array of words. The words are randomly ordered. Write a *recursive class* method named 'find' that has three parameters: an array of strings 'arr' , the search string 's' and an integer starting array index 'lo'. The method returns the integer array index of 's' or -1 if the search word is not found. You may not use loops. Assume that all object references are valid i.e. there no null references.

b) Complete the order-of-growth for the worst case running time for each algorithm.

```
public static boolean isUniqueA(int[] arr) { // N = arr.length
  for(int i =0; i < arr.length -1; i++)
    for(int j=i+1 ; j < arr.length; j++)
      if( arr[i] == arr[j])
        return false;
   return true;
}
```

Worst case running time: _____

```
public static boolean isUniqueB(int[] arr) { // N = arr.length
  for(int i =0; i < arr.length-1; i++) {
    if( arr[i] == arr[i+1])
      return false
  return true;
}
```

Worst case running time: _____

If there no duplicate values then *isUniqueA* & *isUniqueB* must return true i.e. the array values are all unique. However these algorithms make different assumptions about the ordering of the data. If you know that your integer array is already sorted in descending order, which one of the following accurately describes a Computer Scientist's choice of algorithm to check for duplicate values?

A. Use isUniqueA because isUniqueB will be slower.
B. Use isUniqueA because isUniqueB may not find duplicates.
C. Use isUniqueB because isUniqueA will be slower.
D. Use isUniqueB because isUniqueA may not find duplicates.
E. Neither algorithm will function correctly.

Your Answer: _____

3. Recursive lip reading – 14 points (6+6+2).

To lip-read, HAL encodes mouth, tongue and lip movements of human speech using decimal digits. Write a *recursive class* method '*digits'* that takes a single positive integer parameter and returns the sum of digits that are less than 4 (other digits contribute zero) of the parameter. Your method may not use any loops. Examples: digits(2340) returns 5; digits(9876543211) returns 7 (1+1+2+3). Hint: % operator and integer division may be useful.

To look for the pauses in speech, HAL checks for two neighboring digits that are both equal to nine. Write the recursive class method *gap* that takes an int parameter and returns a boolean. Return true if and only if the parameter includes two consecutive nine digits. For example, gap(99) and gap(399391) return true, but gap(949), gap(0), gap(9) return false.

Determine the worst case running time for the *gap* method as a function of the number of digits to be analyzed. Give your answer in big-O notation. You only need to write the order of growth; you do NOT need to explain how you got your answer.

Your Answer: _____

4. [Algorithm Analysis – 15 points (3 points each)].

For each method below determine the order of growth of the *worst case* running time. Write your answer to the right of the method using big-O notation. You only need to write the order of growth; you do NOT need to explain how you got your answer.

(a)
```
public static void threeRows(int[][] arr) {
    int n = arr.length; // N for this problem
    for(int i = 0; i < n; i ++)
      for(int j=0; j < 3; j++)
        System.out.println("arr["+i+"]["+j+"]=" + arr[i][j]);
    }
```

(b)
```
public static int quickCount(int[] arr, int start) {
    int n = start; // N for this problem
    int count = 0;
    while(n>0) {
       count += arr[n];
       n = n / 2;
       if(count > 1000) return count;
     }
     return count;
}
```

(c)
```
public static void crisscross(int size) {
    int n = size; // N for this problem
    for(int i = 0; i <size; i++) {
        for(int j = 0; j<size; j++)
           System.out.print( (i+j)%2 == 0 ? "X" : "O");
        System.out.println();
    }
}
```

(d)
```
// N = initial value of hi-lo+1
public int foo(int[] arr, int key, int lo, int hi) {
        if(lo > hi) return -1;
        if(arr[hi] == key)
             return hi;
         return foo(arr,key,lo , hi-1);
}
```

(e)
```
// N = array.length
public void bar(int[] array) {
    for (int i = 0; i < array.length - 1; i++)
        for (int j = i; 0 <= j; j--) {
           if (array[j] > array[j + 1])  swap(array,j,j+1);
          else j= 0;
        }
};
```

// assume swap(array, x,y) swaps x^{th} y^{th} *array* elements i.e. a constant time operation.

5. Sorting Algorithms – 12 points (6+5+1).

(a) Complete the 6 missing entries in the following table to indicate the order of growth in running time of each sorting algorithm studied in CS125. Use Big-O notation:

Name of Sorting algorithm	Best case	Worst case
?	$O(N^2)$	?
?	?	$O(N \log N)$
Quicksort	?	?

(b) Complete the following method to implement a recursive selection sort. Assume *findMin* and *swap* are already defined: *findMin*(a,x,y) returns the index of the minimum value of array elements {a[x] ,a[x+1]..., a[y] }; *swap*(array,x,y) exchanges the values of the x^{th} and y^{th} array elements.

```
public static void mysort(int[] arr, _____________________________

  if(    ____________________________________
         return;

  int i = findMin( arr, _____________________________

  swap( arr,   _____________________________

  mysort( arr, _____________________________
}
```

(c) Is the following code an example of a Binary search? QuickSort wrapper? Insertion Sort? MedianOfThree? QuickSort? Partition? FindMin? MergeSort? Merge?

```
public static int f2 (int[] A, int lo, int hi, int p){
  if (hi == lo)                                    Answer:_____________________
    if (A[lo] < p)  return lo;  else return lo-1;
  else if (A[lo] <= p)
    return f2 (A, lo+1, hi, p);
  else{
    swap(A, lo, hi);
    return f2 (A, lo, hi-1, p);
  }
}
```

6. Festive Tree – 15 points.

Complete the three class methods (Utility.*prc*, Festive.*main*, Festive.*tree*) below to print an ASCII pine tree. You are **not** permitted to use loops in the *prc* or *tree* methods. The recursive method *tree* must use Utility class *prc* method for any text output. Your program will print trees of different heights, inverted or normal depending on the user's responses. Note the height of the tree does not include the row with the star.

height=1, inverted=false	height=2, inverted=false	height=3, inverted=false	height=4, inverted=false
			
height=1, inverted=true	**height=2, inverted=true**	**height=3, inverted=true,**	**height=4, inverted=true**
			

```
public class Utility {
/** Prints the character 'c' 'count' times then followed by a newline if 'newline' is true. */
      public static void prc(char c, int count, boolean newline) {









      }
}
```

```
public class Festive {

// Add height input logic, then call your recursive tree method
// If the user enters a negative height, redisplay the height question - "height?"
// and obtain a new response from the user.
// Once the user has entered a zero or positive height integer value, print the tree and exit.
public static void main(String[]arg) {
        System.out.println("Upside down tree? Y/N");
        boolean inverted =TextIO.getlnBoolean();









} // end of main
// write your tree() method here:
```

7. Sorting – 12 points (4+4+4).

(a) HAL unwinds after a long day by sorting the world's greatest chess masters by their age. The unsorted data is shown in the array below, indexed from 0 to 11

95	23	60	48	89	16	45	19	82	44	61	31

If HAL passes this array into *selectionSort*, as well as passing 0 and 11 as the initial values of *lo* and *hi*, what are the contents of the array just as the 6th recursive call is starting? Count the initial call, when passed 0 and 11 as arguments, as the start of the first call.

(b) If HAL passes the original array (95,23...) into *Mergesort*, as well as passing 0 and 11 as the initial values of *lo* and *hi*, what are the array contents just before the last merge (i.e. the final *Merge* call in the *MergeSort* call)?

(c) HAL completes a recursive Quicksort partition on the array below: Assume the pivot is 48 and is stored somewhere else. Show the array after every time an element has been placed into one of the partitions. Indicate the current bounds of each partition with [ ] brackets. If a section of array does not change from one row to the next, to save time you may circle that part of the array instead of copying the values. Finally, indicate where the pivot value would be swapped in. You may not need all of the rows.

[22	52	84	89	41	50	38	60	83]

8. Don't quote me– 12 points (5 + 7)

a) Add a public copy constructor to the following Quotes class. The copy constructor will perform a deep copy of the String array: However, to reduce memory requirements do not create additional String objects. You may use a loop in the constructor code.

```
public class Quotes {
 private String[] quotes;
 public int count() { return quotes.length;}
 public String get(int i) { return quotes[i];}
 public Quotes(String[] arr) { this.quotes = arr;}
 public Quotes(Quotes q) {
 // Write your code here




}
```

b) Create a class 'ActorQuotes' that extends Quotes.

- Add a private data member 'name' of type String and public method *getName*.
- Provide a constructor which initializes the Quotes superclass to an empty String array and the private member to the constructor's parameter.
- Provide a copy constructor which performs a copy of the source object using the super classes' copy constructor and copies the reference to the name field.

END OF THE EXAM

OVERFLOW – Use this page if you need more space

OVERFLOW #2– Use this page if you need more space

Scratch paper NetID:_____