**Honors Computer Science I – Final Exam**

**Date: 4/26/11**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1) (6 pts) Consider creating an array of size 100 of pointers to the type struct box. Create this declaration, and then allocate the memory for 100 struct box variables for each of those pointers to point to. No need to initialize any of the fields of each struct. Thus, there’s no need to know the composition of a struct box to finish this question. You may choose the name of your array.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) (4 pts) Write a segment of code to free the memory allocated in the previous question. Please use the same array name you chose in the previous question.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) (4 pts) Give the values of each of the expressions below:

a) 71 ^ 44 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

b) 123 | 78 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

c) 38 & 53 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

d) ~123 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

4) (5 pts) Convert 3FE16 to both base 10 and base 2.

Base Ten: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Base Two: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5) (5 pts) The function compute is defined below. What value does the function call compute(352921114) return?

int compute(int n) {

if (n < 10)

return n;

return n%10\*compute(n/10);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6) (10 pts) Consider rewriting a recursive binary search so that instead of always searching exactly halfway in between the search boundaries, a random element is searched within the search boundaries. Everything else about the algorithm remains the same. (Namely, if the element examined is too small, we recursively search in the right side of the array, or if it’s too big, we recursively search in the left side of the array.) Write a function that implements this recursive search. The prototype is given for you below:

// Returns 1 if searchval is found in array[low…high], and 0

// otherwise. array is sorted in increasing order.

int search(int array[], int low, int high, int searchval) {

}

7) (5 pts) An algorithm runs in O(nk) time, where k is a positive integer. With an input size of n = 100, the algorithm takes 15 milliseconds. With an input size of n = 500, the algorithm takes 1.875 seconds. What is the value of k?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8) (10 pts) The combination problem is very similar to the permutation problem: Given two integers, n and k, with n ≥ k, print out all distinct combinations of k integers from the set {1, 2, 3, …, n}. For example, for n = 4 and k = 2, the following output is produced: (1,2), (1,3), (1,4), (2, 3), (2,4), and (3,4). Complete the function below so that it accomplishes this task. (You have four lines of code to complete, indicated by the underlines.) Note, the initial call to this function takes in an array inSubset[] that is initialized with all 0s, indicating false for each element. In particular, inSubset[k] stores 1 if k is in the current combination being formed and stores 0 if k is not in the current combination being formed.

void combinations(int n, int r, int inSubset[], int length) {

if (r == 0) {

int i;

for (i=0; i<length; i++)

if (inSubset[i])

printf("%d ", i);

printf("\n");

}

else if (n < 0) {

return;

}

else {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

9) (5 pts) What is the worst-case run-time of the following function in terms of the input parameter n? Please give your answer in terms of Big-Oh notation. In order to receive credit, your answer must be a tight upper bound. Thus, if O(n2) is a valid answer, O(n3) will not receive any credit, even if, technically, all algorithms that are O(n2) are also O(n3). Give a justification for your answer.

int function(int array[], int n) {

int i;

while (i < 100) {

int guess = array[rand()%n];

if (guess%10 == 0)

return guess;

i++;

}

return 15;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10) (5 pts) Determine the value of the following sum: .

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

11) (10 pts) Show the final state of an AVL Tree after the following inserts have been made (in this order shown):

50, 30, 10, 20, 3, 14, 25, 40, 70, 43, 77

Note: Credit will only be given for the final state of the tree. Thus, it’s possible that a cascading error could create a loss of many points on this question. Make sure to carefully follow each rebalancing step.

12) (10 pts) Write a function that takes a pointer to the root of a binary tree and returns the number of nodes in that tree that have exactly 1 child. The struct used for the binary tree and the function prototype are provided for you below.

struct tree\_node {

int data;

struct tree\_node \*left;

struct tree\_node \*right;

};

int numNodesWithOneChild(struct tree\_node\* root) {

}

13) (10 pts) Write a function that takes a pointer to the front of a linked list, rearranges the pointers so that the list is reversed (without allocating any new memory OR freeing any old member), and returns a pointer to the new front of the list. If the input parameter is NULL, NULL should be returned. The struct and function prototype are provided below. Feel free to add extra functions. You may define the prototypes of those extra functions as you wish, but you can’t change the prototype already provided.

struct node {

int data;

struct node \*next;

};

struct node\* reverse(struct node\* front) {

}

14) (10 pts) In class we covered the Partition algorithm which is used in Quick Sort. Please fill in the code below so that the function partitions in the input array according to the algorithm.

// Pre-condition: low and high are valid indexes into values

// Post-condition: Returns the partition index such that all the values

// stored in vals from index low to until that index are

// less or equal to the value stored there and all the values

// after that index until index high are greater than that

// value.

int partition(int\* vals, int low, int high) {

int temp;

int i, lowpos;

i = low + rand()%(high-low+1);

swap(&vals[low], &vals[i]);

lowpos = low;

low++;

/\*\*\* FILL IN THE ENTIRE CONTENTS OF THE LOOP \*\*\*/

while (low <= high) {

}

swap(&vals[lowpos], &vals[high]);

return high;

}

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

15) (1 pt) After which hair product wearing real estate tycoon is Trump Towers named?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Scratch Page – Please clearly mark any work on this page you would like graded.**