**Honors Computer Science I – Exam #1**

**Date: 2/3/ 2011**

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

1) (6 pts) Consider executing a binary search on the array below of the number 14. List which indexes of the array are visited before the value 14 is found in index 9.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Value | 2 | 4 | 5 | 6 | 8 | 10 | 11 | 12 | 13 | 14 | 18 | 21 | 26 | 33 |

\_\_\_\_\_\_ , \_\_\_\_\_\_ , \_\_\_\_\_\_ , \_\_\_\_\_\_ , \_\_\_\_\_\_ , \_\_\_\_\_\_ , \_\_\_\_\_\_ , \_\_\_\_\_\_

(Note: More blanks than necessary are provided above intentionally.)

2) (10 pts) Write a **recursive** function that returns 1 (true) if the characters in the array word from index start to index end, inclusive, forms a palindrome, and 0 (false), otherwise. Note: A palindrome is a string that reads the exact same forwards and backwards. For the purposes of this question, a palindrome is case sensitive. Thus, “Madam” is NOT a palindrome.

int isPal(char word[], int start, int end) {

}

3) (10 pts) Write a **recursive** function that returns the average of the numbers in an array. In particular, notice that if we have n numbers a1, a2, …, an, the average of these numbers can be represented as . Specifically, fill in the prototype below so that it returns the average of the first length items in array.

double average(double array[], int length) {

}

4) (10 pts) Convert 81639 to base 5. Put a box around your final answer.

5) (5 pts) Convert A96D16 to binary. Put a box around your final answer.

6) (10 pts) An array can be used to store information about strands of DNA. Specifically, a strand of DNA can be represented as a string of the letters ‘A’, ‘G’, ‘C’, and ‘T’. Unfortunately, these strands turn out to be too long to store in a statically allocated array, thus, the memory for these arrays has to be allocated dynamically. The struct that stores information about a strand of DNA is as follows (length just stores the length of the strand and sequence is to be dynamically allocated:

struct DNAstrand {

char\* sequence;

int length;

};

Write a segment of code that reads in a number representing the size of a strand of DNA, followed by all the letters in the strand into a variable of type struct DNAstrand with the name my\_dna. Assume that you are reading in the information from the file pointed to by ipf in the code segment below and that the file has the number on the first line and the letters (only) on the second line without spaces. Just use scanf with a %c code to read in each individual character. (Note: You’ll have to read in the ‘\n’ character after the number before reading in the letters.)

FILE\* ifp = fopen(“dna.txt”, “r”);

7) (15 pts) Write a function that takes in a pointer to a struct DNAstrand (guaranteed to store a strand of length 1000 or more) and a string (guaranteed to be fewer than 20 characters and contain only ‘A’, ‘G’, ‘C’ and ‘T’ characters) and returns 1 if the short string is a substring of the DNAstrand, and 0 otherwise. Fill in the prototype given below.

int substring(struct DNAstrand\* dna, char snip[]) {

}

8) (10 pts) Write a function that takes a pointer to the front of a linked list and returns the largest value stored in the linked list.

struct ll {

int data;

struct ll\* next;

};

int getMax(struct ll\* list) {

}

9) (15 pts) Let the sequence A1, A2, A3, … defined as follows:

A1 = 2, A2 = 5, An = 5An-1 – 6An-2, for all integers n > 2.

Using regular recursion, writing a function that calculates An for a given n runs slow. For 10 points, write a regular recursive function that takes in the value of n and returns An. For full credit (15 points), utilize memoization to speed up the typical recursive function. The array (of size 21) to be used for the memoization is declared and initialized for you below.

int memvals[] = {0, 2, 5, -1, -1, -1, -1, -1, -1, -1, -1,

-1, -1, -1, -1, -1, -1, -1, -1, -1, -1};

int computeA(int n) {

}

10) (8 pts) Consider the recursive function defined below. What is the output created by the function call print(4)?

void print(int n) {

if (n > 0) {

print(n-1);

print(n-1);

printf("%d ", n);

}

}

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

11) (1 pt) After which former UCF President is Colburn Hall named?

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

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