**2013 SI@UCF Algorithms in C Sharp Test #1**

**Name: Key**

**Date: 7/15/2013**

1) (15 pts) What is wrong with this flood fill function:

static boolean[,] visited;

static char[,] grid;

static int[] dr = {-1,0,1,0};

static int[] dc = {0,1,0,-1};

static int floodFill(int r, int c)

{

 if(r<0||c<0||r>=grid.GetLength(0)||c>=grid.GetLength(1)) return 0;
 if(visited[r,c]) return 0;

 if(grid[r,c]=='#') return 0; // wall

 *visited[r,c] = true; // here*

 int ret = 1;

 for(int delta = 0; delta < 4; delta++) {
 ret+=floodFill(r+dr[delta],c+dc[delta]);

 }

 **visited[r,c] = true; This is out of place. It needs to be put above the for loop that does the recursive calls.**

 return ret;

}

2) (10 pts) Write a recursive function that takes a positive number and returns the number of bits that are on in that number. **Hint if a number %2==1 than the last bit is on**.

int f(int n)

{

 if(n==0) return 0; // base case

 // If the last bit is on than (n%2) is 1, otherwise its 0.

 // When we divide by 2, it removes the last bit.

 return (n%2) + f(n/2);

}

// Alternate solution that uses bitwise operators.

int f(int n)

{

 if(n==0) return 0;

 return (n&1) + f(n>>1);

}

3) (10 pts) Find the maximum contiguous subsequence sum in the following sequence of numbers. Put a box around the contiguous subsequence with the maximal sum as well.

List: 10, 4, 3, -15, 2, 5, -8, 3, -7, **1, 2, 3, -4, 2, 3, -6, 12, 2, -6, 9**, -2, -1

**Just before this sequence the sum is negative, which resets the value to zero.**

Maximum Contiguous Subsequence Sum = \_\_**18**\_\_\_\_\_\_

4) (20 pts) Fill in the following method so that is multiplies the two input matrices and returns the result in a new two dimensional array. (Note: You can figure out the dimensions of the result by looking at the dimensions of both of the input matrices.) You may assume that the input arrays are such that the two corresponding matrices can be multiplied.

public static int[,] add(int[,] m1, int[,] m2)

{

**int n = m1.GetLength(0), m = m1.GetLength(1), p = m2.GetLength(1);**

 **int[,] res = new int[n, p];**

 **for(int i = 0; i < n; i++)**

 **for(int j = 0; j < p; j++)**

 **for(int k = 0; k < m; k++)**

 **Res[I,j] += m1[I,k] \* m2[k,j];**

 **return res;**

}

5) (15 pts) What is the output when f(4) is called? The method f is defined below.

public static int f(int n)

{

 if (n > 0)

 {

 Console.Write(n+” “);

 f(n-1);

 f(n-1);

 }

}

**4 3 2 1 1 2 1 1 3 2 1 1 2 1 1**

6) (12 pts) What is the output running the first permutation algorithm shown in class that uses a boolean used array to print out all of the permutations of “CAMP”. Please list your items in order on the blanks provided below.

1. **CAMP**  7. **ACMP**  13. **MCAP**  19. **PCAM**

2. **CAPM**  8. **ACPM**  14. **MCPA**  20. **PCMA**

3. **CMAP**  9. **AMCP**  15. **MACP**  21. **PACM**

4. **CMPA**  10. **AMPC**  16. **MAPC**  22. **PAMC**

5. **CPAM**  11. **APCM**  17. **MPCA**  23. **PMCA**

6. **CPMA**  12. **APMC**  18. **MPAC**  24. **PMAC**

7) (15 pts) Add memoization to the recursive method shown below:

**int[,] memo;**

public static int f(String s, String t) {

 if (s.Length == 0 || t.Length == 0) return 0;

 **if(memo[s.Length, t.Length] != -1)**

 **return memo[s.Length, t.Length];**

 if (s[s.Length-1] == t[t.Length-1])

 {

 **int res = 1+f(s.Substring(0,s.Length-1),**

 **t.Substring(0,t.Length-1));**

 **memo[s.Length, t.Length]=res;**

 **return memo[s.Length, t.Length];**

 }

 int ans1 = f(s.Substring(0, s.Length-1), t);

 int ans2 = f(s, t.Substring(0, t.Length-1));

 **memo[s.Length,t.Length] = Math.min(ans1,ans2);**

 return Math.Min(ans1, ans2);

}

8) (3 pts) For what breakfast food is Dunkin Donuts known? **\_Donuts­­­­­\_\_\_\_\_\_\_\_\_\_\_ \_**