# Computer Science Foundation Exam

December 15, 2006

## Computer Science

### Section 1B

<table>
<thead>
<tr>
<th></th>
<th>Max Pts</th>
<th>Type</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>8</td>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td>8</td>
<td>DSN</td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>12</td>
<td>DSN</td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td>14</td>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Q5</td>
<td>8</td>
<td>DSN</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SOLUTION KEY**
1. [8 pts] Using summations find the value of $beta$ in terms of $n$ after the following segment has been executed.

```c
beta = 0;
alpha = 4;
for ( k= 1 ; k < 2 ; k++) {
    for ( j = n; j <= 2n ; j++)
        beta = beta + alpha *j - 6*n ;
}
```

$$beta = \sum_{j=n}^{2n} 4j - 6n \quad \quad \quad \text{[step1]}$$

$$= \sum_{j=n}^{2n} 4j - \sum_{j=n}^{2n} 6n \quad \quad \quad \text{[step2]}$$

$$= \sum_{j=1}^{2n} 4j - \sum_{j=1}^{n-1} 4j - 6n(2n-n+1) \quad \quad \text{[step3]}$$

$$= 4(2n)(2n+1)/2 - 4(n-1)n/2 - 6n(n+1) \quad \quad \text{[step4]}$$

$$= 8n^2 + 4n - 2n^2 + 2n - 6n^2 - 6n \quad \quad \text{[step5]}$$

$$= 0$$

**Grading:**

- **step1**: 2 points
- **step2**: 1 point
- **step3**: 3 points
- **step1**: 1 point
- **step1**: 1 point
2. [8 pts] For a hypothetical application, it is necessary to create and maintain a oneway-linked list \textit{alpha}. Each node in the list \textit{alpha} consists of an integer data element, value, a link to a binary tree, \textit{btree}, and a link to another oneway-linked list named, \textit{segment}. Each node of the binary tree, \textit{btree}, contains an integer value, \textit{gamma}, and a name, \textit{sales}, no more than 30 characters in length. Each node of \textit{segment} contains an item named, \textit{rate}, of type double. Declare the node structures for \textit{alpha}, \textit{btree} and \textit{segment} in their proper compilation order.

```c
struct segment{
    double rate;
    struct segment *next;
}; [2 pts]

struct btree{
    int gamma;
    char sales[30];
    struct btree *left,*right;
}; [3 pts]

struct alpha{
    int value;
    struct btree *treenext;
    struct segment *segnext;
    struct alpha *nodenext;
}; [3 pts]
```

[take of 2 points if the declaration order is wrong]
3. [12 pts] Consider a linked list M with following node structure:

```c
struct node {
    int value;
    char names[30];
    struct node * next;
};
```

The list is a sorted list, such that all names are in alphabetical order. Write a function `newname` in ‘C’ to insert a new node in its proper place in the list.

```c
struct node *newname(struct node *M, int dinteger, char dname[30])
{
    /*1. Declare local variables and then save the pointer to head in a new variable. */
    struct node *original, *pNew, *cur;
    original = M;          [1 point]

    /*2. Create a new node for the linked list and copy data into it */
    pNew = (struct node*)malloc(sizeof(struct node));
    strcpy(pNew->firstname, first);
    pNew->value = dinteger;
    strcpy(pNew->names, dname);   [1 point]

    /* 3. If the list is empty, change head to the new node, return this head. */
    if (M == NULL){
        M = pNew;
        return M;
    }        [2 points]

    /*4. If new node is alphabetically smaller than the first node, put new node at head, return this head.*/
    if (strcmp(dname, M->names) < 0){
        pNew->next = M;
        M = pNew;
        return M;
    }            [2 points]
}
/*5. Otherwise, traverse through list until it finds alphabetically ordered spot for new node. */
while (M!=NULL) {
    if (strcmp(dname, M->names) > 0){
        cur = M;
        M = M->next;
    }
    else if (strcmp(dname, M->names) < 0)
        break;
}
/* 6. Add new node in spot, return original head. */

cur->next = pNew;
pNew->next = M;
return original;
4. [14 pts]
(a) [6 pts] Trace the following function for the call `gain( 2, 7)`. Show your work.

```c
int gain(int a, int n) {
    if(n == 1)  return a;
    if ( n %2 == 0)
        return gain( a , n/2) * gain (a, n/2);
    else
        return gain( a , n/2) * gain (a, n/2)* a;
}
```

gain(2,7) = gain(2,3) * gain (2,3) * 2   (1)
gain(2,3) = gain (2,1) *gain(2,1) * 2
    
    = 2 * 2 * 2 = 8

gain(2,7) = 8 * 8 * 2 = 128    [6 points]  

[partial points may be awarded]

(b) [8 pts] Write the recurrence relation for the above function and solve it to work out
the maximum possible number of operations needed by this function in terms of n. Show
all the steps. You are not allowed to use the Master’s theorem to solve the recurrence
relation.
The maximum number is obtained by considering line 7

```
T(n) = 2 T(n/2) + 4
T(1) = 1
```

The constant number is not very crucial, students may also
simply consider the multiplications in line 7 and come up with

```
T(n) = 2 T(n/2) + 2
T(1) = 1
```

which is also correct

[3 pts for correct recurrence relations]
Solution:

\[ T(n/2) = 2 \cdot T(n/4) + 4 \]
\[ T(n) = 4 \cdot T(n/4) + 2(4) + 4 \]
Or \[ T(n) = 4 \cdot T(n/4) + 3(4) \]

Now \[ T(n/4) = 2 \cdot T(n/8) + 4 \]
So \[ T(n) = 8 \cdot T(n/8) + 4(4) + 3(4) \]
Or \[ T(n) = 8 \cdot T(n/8) + 7(4) \]

This can also be written as
\[ T(n) = 2^3 \cdot T(n/2^3) + (2^3-1) \cdot (4) \]

[ 2 pts for correctly reducing the Right side ]

**General case**

\[ T(n) = 2^k \cdot T(n/2^k) + (2^k-1) \cdot (4) \]

[ 1 pts for correct general case ]

Let \( n/2^k = 1 \)
\[ 2^k = n \]
\[ k = \log n \]

[ 1 pts for correctly working out value of k ]

Substituting the values in general case

\[ T(n) = n \cdot T(1) + 4(n - 1) \] [ 1 pt for correct substitution ]
\[ T(n) = n + 4n - 4 \]
\[ T(n) = 5n - 4 \]
5. [8pts] Develop a RECURSIVE function which prints the digits of an integer \( num \), one digit on every new line, in the order they appear in \( num \). Thus given the integer 7354, it should print:
7
3
5
4

Note, you can not convert the given integer to a string format.

[ 8 pts for correct solution, Only 2 pts if solution prints digits in reverse order ]

[ 4 pts for a non-recursive solution]

```c
void printdigits( int num)
{
    if (num == 0)
        return 0;
    else
    {
        printdigits( num/10);
        printf(" %d \n", num % 10);
    }
}
```