

# CS13002 Programming and Data Structures, Spring 2005

## Class test 2

Total points: 30

April 05, 2005

Total time: 1 hour

---

Roll no: \_\_\_\_\_ Name: \_\_\_\_\_ Section: \_\_\_\_\_

*Write your answers in the question paper itself. You may use extra blank sheets for rough work, but your answers must fit in the respective spaces provided. Answer all questions.*

1. Suppose that we have got a two-dimensional array  $A = (a_{ij})$  with  $m$  rows and  $n$  columns. Let the one-dimensional row-major representation of  $A$  be stored in the array  $B = (b_i)$  of size  $mn$ . Assume that indexing in the arrays is zero-based.

(a) Given a pair of indices  $i, j$  for the matrix  $A$ , the element  $a_{ij}$  is stored as  $b_k$ , where (5)

$k =$  \_\_\_\_\_ (in terms of  $i, j, m, n$ ).

(b) Given an index  $k$  in the array  $B$ , the element  $b_k$  corresponds to the element  $a_{ij}$  in  $A$ , where (5)

$i =$  \_\_\_\_\_ and  $j =$  \_\_\_\_\_ (in terms of  $k, m, n$ ).

2. Let  $A$  be a square ( $n \times n$ ) matrix. We want to compute the matrix  $B = A - A^t$  and store this matrix in  $A$  itself. Here  $A^t$  denotes the transpose of the matrix  $A$ . Write a function that accepts  $A$  and computes  $B$  without using an additional array. You are not allowed to use any extra variables other than two indices  $i$  and  $j$ . Assume that **ROWDIM** and **COLDIM** are the storage dimensions and  $n \times n$  is the actual dimension of  $A$ . Note that if  $A = (a_{ij})$  and  $B = (b_{ij})$ , then  $b_{ij} = a_{ij} - a_{ji}$  and  $b_{ji} = a_{ji} - a_{ij}$  for all indices  $i, j$ . (10)

```
void matFunc ( int A[ROWDIM][COLDIM] , int n )  
{  
    int i, j;
```

```
}
```

3. You are given a linked list. Your task is to create two new linked lists, the first of which should contain the 1st, 3rd, 5th, . . . elements and the second the 2nd, 4th, 6th, . . . elements of the input list. The following code segment provides a solution. Fill in the blanks to complete the segment. Evaluation of your answer will depend on overall correctness.

The function `createLists` assumes that there is a dummy node at the beginning of each list (input as well as output). The input list is headed by the pointer `head`. Odd-numbered elements are to be stored in the list headed by the pointer `oddhead`, and the even-numbered elements are to be stored in the list headed by the pointer `evenhead`. Assume that the input pointer `head` already points to a properly allocated list with a dummy node at the beginning. Assume also that both `oddhead` and `evenhead` are already allocated memory only for the dummy nodes. We number the elements of the input list from 1. (10)

```

/* First define a structure for a node in the list */
typedef struct nodeTag {
    int data;
    /* Declare the self-referencing pointer */
    _____ next;
} node;

void createLists ( node *head , node *oddhead , node *evenhead )
{
    node *src, *dest1, *dest2;
    int flag = 1;

    /* Initialize the source and destination pointers to point to the dummy nodes */
    src = head; dest1 = oddhead; dest2 = evenhead;

    /* As long as the source list is not fully traversed */
    while ( _____ ) {
        if (flag == 1) {
            /* Allocate memory for a new node */
            _____

            /* Advance the destination pointer by one node */
            _____

            /* Copy data from source */
            _____
        } else {
            /* Allocate memory for a new node */
            _____

            /* Advance the destination pointer by one node */
            _____

            /* Copy data from source */
            _____
        }
        src = src -> next;
        if (flag == 1) flag = 2; else flag = 1;
    }
    dest1 -> next = dest2 -> next = NULL;
}

```