CS11001 Programming and Data Structures, Autumn 2010

Class Test 2

Maximum marks: 30 | November 09, 2010 | Total time: 1 hour

Roll no: 10FB1331 **Name:** Foolan Barik **Section:** @

Write your answers in the question paper itself. Be brief and precise. Answer all questions.

1. A paragraph is stored in a two-dimensional array of characters. Each line of the paragraph is stored in a row of the array as a null-terminated string. An empty line indicates the end of the paragraph.

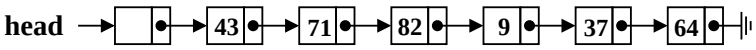
(a) Complete the following function that, given a paragraph stored in the format mentioned above, prints the paragraph line by line, and stops when the terminating empty line is encountered. **(4)**

```
void prnPara ( char para[][MAXLEN] )
{
   int i; /* Do not use any other variable */

   /* Print each line as a string until a blank line is found */

   for (i=0; strlen( para[i] ); ++i) printf("%s\n", para[i] );
}
```

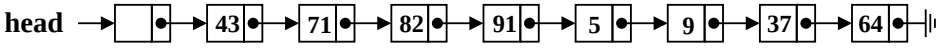
(b) Complete the following function that right justifies a paragraph with respect to a target length T as follows. For each line in the paragraph, its length l is computed. If $l > T$, then right justification fails. If $l \leqslant T$, you shift the line by $T - l$ positions to the right, so that the line now ends at exactly the index $T - 1$ (excluding the terminating null character). The first $T - l$ positions are then filled with spaces. The right justification loop terminates whenever an empty line is found (indicating the end of the paragraph). **(12)**

```
void rightJustify ( char para[][MAXLEN], int T )
{
   int i, j, len; /* Do not use any other variable */

   /* Repeat so long as an empty line is not found */

   for (i=0; strcmp( para[i] , "" ); ++i) {
      /* Compute in len the length of the i-th line */

      len = strlen(para[i]) ;
      if (len > T) {
         printf("Line %d is too wide to fit in %d characters\n", i, T);

         continue; /* Skip the rest of the current iteration */
      }
      /* Write a loop to shift characters by T-len positions to the right */

      for (j=len-1; j>=0; --j) para[i][j+(T-len)] = para[i][j];
      /* Write a loop to fill the first T-len characters by spaces */

      for (j=0; j<T-len; ++j) para[i][j] = ' ';
      /* Terminate the i-th line by the NULL character */

      para[i][ T ] = '\0' ;
   }
}
```

2. Assume that you are given a linked list with an even number of nodes (excluding the dummy node at the beginning). Your task is to write a function in order to locate the middle of the list, and to subsequently insert two new elements in that middle position. For the sake of ease of programming, you may assume that a dummy node is maintained at the beginning of a linked list. Here is an example of a linked list with six elements $43, 71, 82, 9, 37, 64$. After inserting two new elements 91 and 5 at the middle, the list changes to $43, 71, 82, 91, 5, 9, 37, 64$.



(a) A linked list



(b) After insertion of two elements at the middle

In order to locate the middle of the list, we use two pointers p and q. They are initialized to point to the first node of the list (the dummy node). Subsequently, in a loop, p advances down the list by two cells, whereas q advances down the list by one cell. When p reaches the end of the list, the pointer q is ready for insertion at the middle. Do not allocate two cells together in a single memory allocation call, since if you do so, you cannot free a cell individually. **(14)**

```
typedef struct _node {
    int data;

    struct _node *next;
} node;

void insertMid (node *head, int n1, int n2 )
{
    node *p, *q; /* Do not use any other variable */

    p = q = head;
    /* While the middle of the linked list is not located */

    while ( p -> next != NULL ) {

        p = p -> next -> next ; /* p advances by two cells */

        q = q -> next ; /* q advances by one cell */
    }
    /* Here, q points to the node after which insertion will be made */
    /* Allocate memory to the first new node */

    p = (node *)malloc(sizeof(node)) ;
    /* Allocate memory to the second new node */

    p -> next = (node *)malloc(sizeof(node));
    /* Set the data fields */

    p -> data = n1 ; p -> next -> data = n2;
    /* Establish link from the second node created */

    p -> next -> next = q -> next;
    /* Establish link from the list to include the new nodes */

    q -> next = p;
}
```