Helwan University

Faculty of Computers and Information

Review Problems 2

Department of Computer Science
Data Structures

For all of the following problems, when it is stated "in the user level" you should assume that you only have the object code of the data structure along with the interface given in the header file. On the other hand, When it is stated that "in the implementation level" you should assume that you have the source code file of the data structure and you should write the desired piece of code within that source code file.

1. How to use a general list either as a queue or a stack?
2. Write a function that returns a copy of a particular element in a general list. (in both levels).
3. Write the function `void JoinList(List *pl1, List *pl2)` that copies all entries from `l1` onto the end of `l2`; `l1` remains unchanged, as do all the entries previously in `l2`. The list `l2` must be, at least, created. (in both levels)
4. Write the type definition of a doubly-linked-list node `DLLNode`. Each node has a data entry field and two pointers, one points to the next node and the other points to the previous one. Define the doubly linked list to be, simply, a pointer to a node.
   (a) Write a function to initialize the list.
   (b) Write an iterative function to insert an element at a particular position.
   (c) Rewrite the previous function recursively.
5. Write the function `void DestroyNodes(ListNode *)` recursively (implementation level). Rewrite it iteratively; which version is more efficient?
6. Consider the type definition of a general list studied in lectures (a `struct` of some fields). Rewrite the function `DestroyList` recursively. This can be done by calling the function `void DestroyNodes(ListNode *)` written above. What is the reason that prevents us from writing the recursive function `DestroyList` directly?
7. It is required to keep the data of employees in a general list. Write the type definition `ElementType` such that each employee will have the following data.

   -Name　　-Home Address　　-Date of Birth　　-Company (Name, Address, Phone number)

   Note that, any date should include day, month, and year; any address should include street, city, and zip code.
8. The C programming language does not support generic data types; we have to do some extra work to insert more than one data type in the same structure! Assume that our application requires inserting either an employee or a student in the same structure. Assume that a student has both, name and home address, as an employee. However, he does not have a company field; rather, he has an institution field. Every Institution must have name, address, departments (up to 20), and degrees granted (up to 5 degrees). Rewrite `ElementType` such that it can accommodate either a student or an employee. Note that, an extra field, e.g., `etype`, is needed to allow the user to identify, e.g., when retrieving an element from the data structure, which type the element has. For example:

```
DeleteList(&l, pos, &e);
switch (e.etype){
    ...
}
```

   Hint: Use `union` instead of `struct` to combine the fields "company" and "institute". This will reserve, in memory, only the maximum, not the total, of their sizes. What is the size of `ElementType`? Calculate it by hand then check by calling `sizeof`.