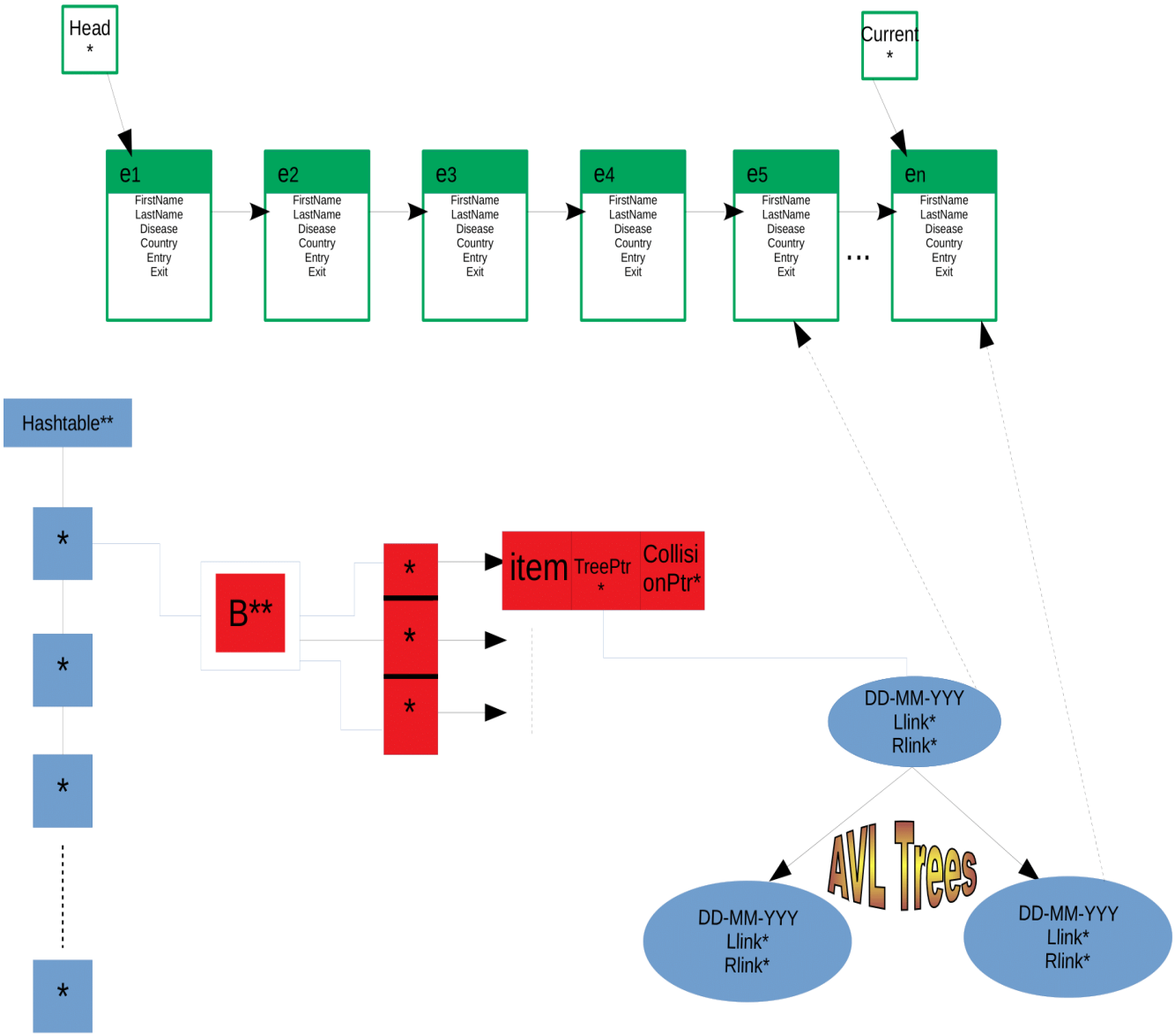
Προγραμματισμός Συστήματος - 1η Εργασία, Εαρινό Εξάμηνο 2020

Δημήτρης Φωτεινός, Τμήμα Πληροφορικής και Τηλεπικοινωνιών AM: 1115201700181

Δομές Δεδομένων:

Παρακάτω, ακολουθεί σχηματική αναπαράσταση των δομών δεδομένων που έχουμε χρησιμοποιήσει και στη συνέχεια αναλυτική επεξήγηση του κώδικα καθώς και παραδοχές υλοποίησης.



Συνδεδεμένη Λίστα:

```c
typedef struct Record{

    patientInfo recordID;
    patientInfo patientFirstName;
    patientInfo patientLastName;
    patientInfo diseaseID;
    patientInfo country;
    patientInfo entryDate;
    patientInfo exitDate;
    struct Record* recordPtr;

} patientRecord;

/*Record of a Patient*/
typedef patientRecord* recordPointer;
```

Αρχικά, έχουμε 2 δείκτες τύπου **recordPointer**. Ο δείκτης Head* , δείχνει στο πρώτο Entry και ο τελευταίος δείκτης Current* δείχνει στο πιο πρόσφατο Entry που διαβάζουμε κάθε φορά απ'το δοσμένο αρχείο.
Παρατηρούμε ότι μέσα σε ένα Entry αποθηκεύονται όλες οι πληροφορίες που μας ενδιαφέρουν. Τέλος, κάθε οντότητα τύπου patientRecord έχει έναν δείκτη σε κάθε επόμενο Entry.

Πίνακας Κατακερματισμού:

```c
typedef struct Entry{

    BigBucket bucket;

} HashEntry;

/*HashTable*/
typedef HashEntry* HashPointer;  //Pointer of a Hashtable
typedef HashPointer* HashTable;  //Pointer pointing into a pointer of the Hashtable
```

Αυτό που στην ουσία έχουμε ορίσει ως πίνακα κατακερματισμού δεν είναι τίποτα άλλο πέρα από έναν διπλό δείκτη ο οποίος δείχνει σε δείκτες που δείχνουν στην οντότητα ενός HashEntry. Εδώ πρέπει να προσέξουμε καθώς υπάρχει κίνδυνος να συγχέουμε την έννοια του HashEntry με αυτην του Bucket. Θα δούμε παρακάτω την διαφορα.
Η οντότητα του HashEntry περιέχει μέσα της έναν διπλό δείκτη της οντότητας ενός BucketEntry. Παρακάτω βλέπουμε το εσωτερικό της δομής ενός BucketEntry.

```c
typedef struct BucketEntry{

    bucketItem item;
    TreePointer treePtr;
    struct BucketEntry** collision;

} Bucket;

/*Bucket of Hashtable*/
typedef Bucket* BucketPointer;      //Pointer for the bucket of the Hashtable
typedef BucketPointer* BigBucket;   //The real "bucket" structure
```

Καλό θα ήταν να ξεκαθαρίσουμε τώρα πως αυτό που στην ουσία αποτελέι ένα Bucket, είναι το σύνολο όλων αυτών των subBuckets, δηλαδή όλων αυτών των δεικτών οι οποίοι δείχνουν στην δομή Bucket. Τον αριθμό των subBuckets μπορούμε να τον βρούμε εύκολα απ΄τον μαθηματικό τύπο: $\frac{bucketSize}{24}$, καθώς το μέγεθος της δομής Bucket , δηλαδή ενός subBucket είναι 24 bytes (3 pointers). Τέλος, κάθε subBucket έχει προφανώς 1 entry είτε αυτό είναι κωδικός χώρας είτε κωδικός ασθένειας, επίσης έχει έναν δείκτη που δείχνει στην ρίζα ενός κόμβου δέντρου τύπου AVL και ακόμη, έναν διπλό δείκτη στην δομή ενός Bucket, στην περίπτωση που γεμίσουν όλοι οι υπο-κουβάδες και έχουμε collision.

Δυαδικό Ισσοροπημένο Δένδρο Αναζήτησης:

```c
typedef struct Tree{

    BalanceFactor BF;
    treeItem item;
    struct Tree* LLink;
    struct Tree* RLink;
    recordPointer patientRecord;


} BalancedTree;


/*Binary Balanced Tree*/
typedef BalancedTree* TreePointer;  //Pointer for the Balanced Binary Tree
```

Η συγκεκριμένη περίπτωση δομής είναι αρκετά τεττριμένη. Η δομή ενός δένδρου αποτελείται από ένα περιεχόμενο εισόδου τύπου treeItem, ενός ακέραιου αριθμού τον οποίο θα μπορούσαμε να τον χαρακτηρίσουμε ως ΄δείκτη ισσοροπίας' που θα μας βοηθήσει στις διάφορες συναρτήσεις που θα υλοποιήσουμε. Τέλος έχουμε 1 δείκτη για το δεξί παιδί του δένδρου και αντίστοιχα 1 δείκτη για το αριστερό και φυσικά έναν δείκτη στο αντίστοιχο Record.

Για να μετταγλωτίσουμε το πρόγραμμα, γράφουμε: make , και έπειτα ./diseaseMonitor -p patientRecordsFile -h1 diseaseHashtableNUmOfEntries -h2 countryHashtableNUmOfEntries -b bucketSize.