Lebanese University
Faculty of Sciences 1

I2204 E – INFO216 E

Imperative Programming

Sesssion 2 2016-2017
Sat. July. 15th am
Duration: 2 hours

Exercise I [30 min, 25 points]

Here is a small program that has some problems (6 to be exact). You should find all the bugs and give a brief explanation of each problem. You are not required to fix the bugs (and there might not be obvious fixes in some cases), but sometimes an easy way to explain a bug is to show how to correct it.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   typedef struct point {
4       int x, y, z;
5   } Point, *PointPtr;
6   PointPtr add(PointPtr a, PointPtr b) {
7       Point res;
8       res.x = a->x + b->x;
9       res.y = a->y + b->y;
10      res.z = a->z + b->z;
11      return &res;
12  }
13  int main(int argc, char* argv[]) {
14      Point a = {1, 2, 3};
15      PointPtr b;
16      b->x = 3;
17      b->y = 5;
18      b->z = 7;
19      PointPtr c = malloc(sizeof(PointPtr));
20      c = add(&a, b);
21      printf("a = (%d, %d, %d)\n", a.x, a.y, a.z);
22      printf("b = (%d, %d, %d)\n", b->x, b->y, b->z);
23      printf("a + b = (%d, %d, %d)\n", c->x, c->y, c->z);
24      free(b);
25      free(c);
26  }
```

To answer, copy and fill the following table:

bug no	line number	bug	explanation
1			
2			
3			
4			
5			
6			

Exercise II [30 min, 25 points]

Consider the following data type:

```
typedef struct node{int d; struct node *next, *prev;} node;
```

Write a function "reverse" which reverses the order of the nodes in a given doubly linked list of integers. For example, the list [3⇌9 ⇌5⇌1] becomes after the call to reverse [1⇌ 5⇌ 9⇌ 3].

Exercise III [60 min, 45 points]

We want to create an index for a given text. We will do it step by step, following questions 1 to 5. First we write a function "createArrayofLL" that takes as a parameter a text file name, creates and returns an array of 26 sorted linked lists, one for each alphabet letter. The function should read the text file and insert each word into one of the lists based on its first letter, eg: any word that begins with 'a' goes, sorted, to the linked list at the entry 0 of the array. Second, we will write the function "writeIndex" which uses the dynamically constructed index to write a new text file containing the index.

Note that there will be no duplicates in the lists, and that the comparison should be case insensitive. Also note that the punctuations and the quotes at the beginning end of the words should not be accounted.

1. Define the appropriate structure "node" to hold a word and a "next" field.
2. Write the function "createArray26" which allocates in the heap an array of 26 pointers to node, initializes them to NUL and returns the array address.
3. Write the function "insertSorted" that insert a given word to its right place in a given linked list. The function should do nothing if the word already exists in the list. The comparison should be case insentisitive.
4. Using the functions in questions 2 and 3, write the function "createArrayofLL". Example : if given the file name "in.txt" in fig. 1 below, the function should construct and fill the index, fig.2, in the heap, then return it.
5. Write the function "writeIndex" which given an index copies its content (words) into a text file. Each set of words corresponding to a letter are on one line, dash separated, the line begins by the index letter followed by two vertical points (:). As an example, if given the index in fig.2, the function creates and fills the text file "out.txt" in fig. 3. As one can note, the letters corresponding to empty linked lists are skipped.

in.txt

Julia was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once she had peeped into the book her sister was reading, but it had no pictures or conversations in it. "And what is the use of a book", thought Julia, "without pictures or conversations?"

fig. 1

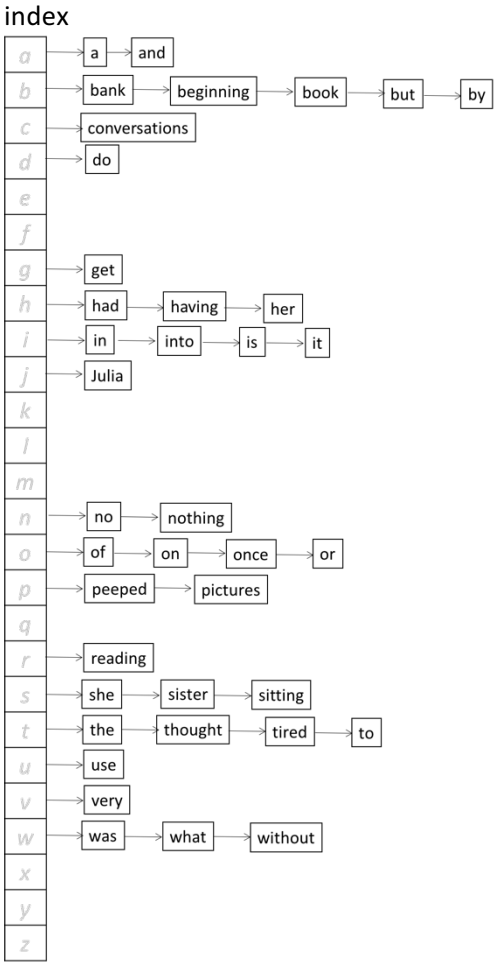

fig. 2

out.txt

a: a - and
b: bank - beginning - book - but - by
c: conversations
d: do
g: get
h: had - having - her
i: in - into - is - it
j: Julia
n: no - nothing
o: of - on - once - or
p: peeped - pictures
r: reading
s: she - sister - sitting
t: the - thought - tired - to
u: use
v: very
w: was - what - without

fig. 3