Lebanese University
Faculty of Sciences 1

INFO216 E

Imperative Programming II

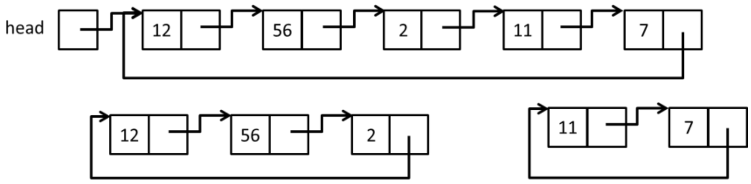
Final 14.15
Mon. Jun 8th am
Duration: 2 hours

Exercise I [20 min – 15 points]

Write the function "split" that splits a given circular linked list into halves and returns their heads through its parameters. If the length of the list is odd, the middle node will be in the first list. Otherwise, both lists have the same length. You are not allowed to delete nodes, nor to create new ones. Also the order the nodes in the list is to be maintained. Below are the definition of the type "node" and the prototype of the function "split".

```
        typedef struct Node{ int d; struct Node *next;} node;
        void split(node *head, node **h1Ref, node **h2Ref);
```

Example:



Exercise II [50 min – 30 points]

We define the type node : `typedef struct Node{ char d; int freq; struct Node *next;} node;`

1. Consider a linked list of "node" where the freq values for all the nodes is 1. Write the function "encode" that condenses each group of successive repeated nodes into one node, assigns the number of repetition to be the value of its "freq" field and deletes the others. No node allocation is allowed in this task.

```
Example:
        before :     head → a,1→ a,1→ a,1→ c,1→ c,1→ d,1→ d,1→ d,1→ d,1→ a,1→ a,1
        after :      head → a,3→ c,2→ d,4→ a,2
```

2. Write the function "decode" that reads a compressed file, uncompresses it and displays it on the screen and returns the compression ratio. The compression ratio is the ratio of the character counts before and after the compression. We suppose that the frequency value does not exeed 9.

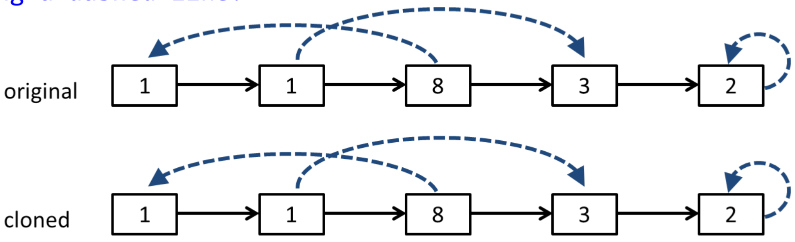
```
Example:
        file content:       a3c2d4a2
        output :            aaaccddddaa
        compression ratio = 11/8= 1.375
```

Exercise III [50 min – 25 points]

We define the type node : `typedef struct Node{ int d; struct Node *next, *arbit;} node;`

Consider a singly linked list with every node having an additional arbitrary pointer, called "arbit". The arbit pointer may be NULL or may point to any node in the list. Write a function "clone" to create an exact copy of this list.

```
Example:      in the figure below we represent the next pointer using a solid line and the arbit
              pointer using a dashed line.
```


original
1
1
8
3
2
cloned
1
1
8
3
2