Lebanese University
Faculty of Sciences 1

I2204 E – INFO216 E

Imperative Programming

Partial 2017-2018
Tue. Nov. 21th pm
Duration: 1 hour

Exercise I: Mysterious enigma (6 points)

Draw the memory state tracing the following program. Deduce the output.

```c
#include <stdio.h>
int enigma(int n1, int n2) {
    if (n1 == 0) return 0;
    return n2 + enigma(n1 - 1, n2);
}
int mystery(int n) {
    if (n == 0) return 1;
    return enigma(2, mystery(n - 1));
}
void mysteryTest(){
    printf("%d",mystery(2));
}
void main(){
    mysteryTest();
}
```

Exercise II: Recursive raising to the power (6 points)

It is easy to implement an iterative function "raiseIntToPower" that computes n raised to the kth power, n and k a positive intergers.

```c
#include <stdio.h>
void raiseIntToPowerTest();
int raiseIntToPower(int n, int k)
{
    int result = 1;
    for (int i = 0; i < k; i++)
        result *= n;
    return result;
}
void main(){
    raiseIntToPowerTest();
}
```

```c
void raiseIntToPowerTest(){
    printf("%d\n",raiseIntToPower(3,1));//3
    printf("%d\n",raiseIntToPower(3,4));//81
    printf("%d\n",raiseIntToPower(3,3));//27
    printf("%d\n",raiseIntToPower(-3,5));//-243
    printf("%d\n",raiseIntToPower(0,4));//0
    printf("%d\n",raiseIntToPower(0,0));//1
    printf("%d\n",raiseIntToPower(5,0));//1
}
```

Rewrite this function so that it operates recursively, taking advantage of the following insight:

- If k is even, n^k is the square of n raised to the power k / 2.
- If k is odd, n^k is the square of n raised to the power k / 2 times n.

In solving this problem, you need to identify the simple cases necessary to complete the recursive definition. You must also make sure that your code is efficient in the sense that it makes only one recursive call per level of the recursive decomposition. In your code you can declare no local variables of any type, you only use the parameters.

Exercise III: Dominos linked list (10 points)

The game of dominos is played with rectangular pieces composed of two connected squares, each of which is marked with a certain number of dots. For example, each of the following five rectangles represents a domino:

Dominos can be connected end-to-end to form chains, subject to the condition that two dominos can be linked together only if the numbers match. For example, you can form a chain consisting of all five of these dominos by connecting them in the following order (notice we rotated some pieces):

1. Define the type node, representing a domino in a list, having two interger fields left and right and one next field.
2. Write the function rotate180 which given a node, rotates the domino it represents 180 degrees, i.e. switches the left and right values.
3. Write the function "formsDominoChain" which given a domino list returns 1 if the nodes in the list form a domino chain, 0 else. The function may rotate nodes to make them fit in the chain.