# 

# DSA Exercise Recursion

# SECJ2013 -02

# 

# Group 6 :

LEE SZE YUAN (A19EC0068)

LOH YEW CHONG (A19EC0076)

CHIA WEI HONG (A19EC0033)

# 

# 

# Pg 66

Question 2

1. 343,

step 1:

problem: fun2(7,3)

step 2: ↓

return 7+ fun2(7,3-1)

step 3: ↓

return 7+ fun2(7,2-1)

step 4: ↓

return 7+fun2(7,1-1)

step 5: ↓

return 1

step 10: final answer

343

step 9: ↖

return 7\*49 = 343

step 8: ↖

return 7\*7=49

step 7: ↖

return 1\*7=7

step 6: ↑

return 1

1. Problem solved by the recursive: multiplying.

Simple solution: return 1

Recursive process: x \* fun2(x,n-1)

Terminal case: (n==0)

# 

# Page 67

**Question 4**

**a)**

Step 1:

GCD(3, 8);

Step 2: ↓

return GCD(8 ,3%8 = 3 );

Step 3: ↓

return GCD(3, 8%3 = 2);

Step 4: ↓

return GCD(2, 3%2=1);

Step 5: ↓

GCD(2, 3%2=1);

Step 6 : ↓

Since if (a%b == 0), will return 1

GCD(2, 1); return 1

Step 7 : ↓

Return the result to sub-problem 1

return 1

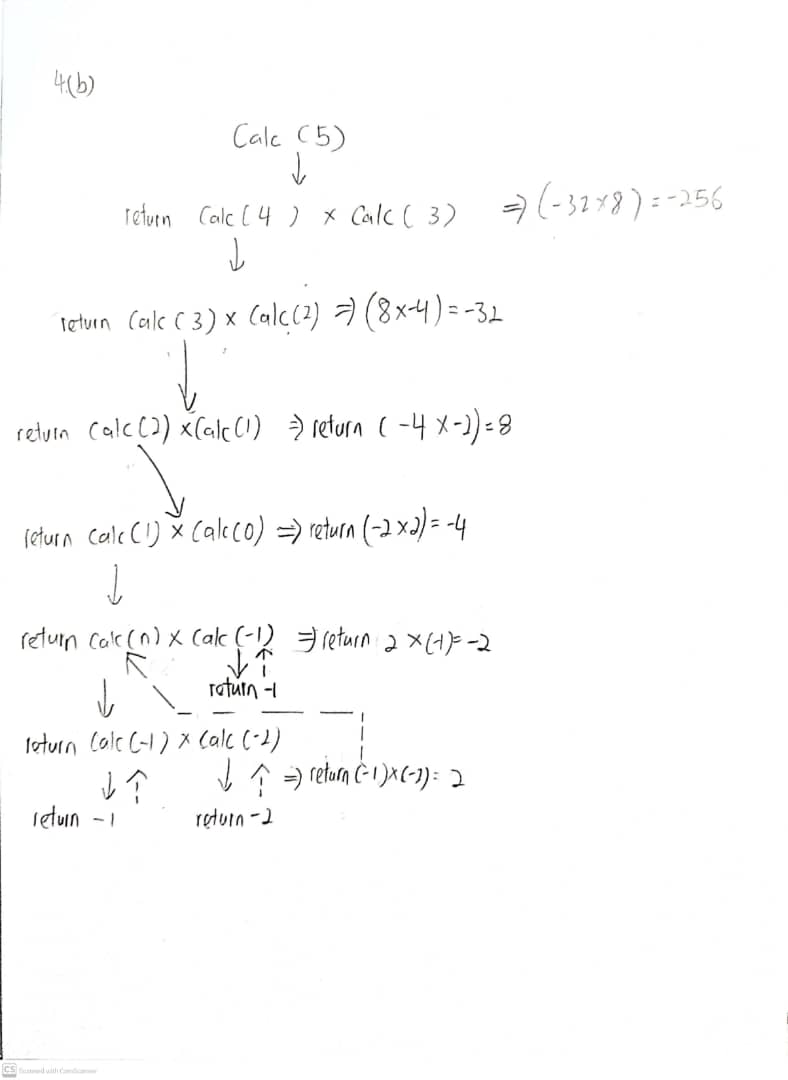
Step 8 : ↓

Return the result to the called function main().

return 1

**Question 4**

**b)**



5: Calc(4)\*Calc(3) return -256

4: Calc(3)\*Calc(2) return -32

3: Calc(2)\*Calc(1) return 8

2: Calc(1)\*Calc(0) return -4

1:Calc(0)\*Calc(-1) return -2

0:Calc(-1)\*Calc(-2) return 2

Calc(-1) return -1

Calc(-2) return -2

# 

# Pg 68

Question 5

1. 218
2. Step 1:

Problem : function\_1(5,15)

Step 2: ↓

return (5\*function\_1(5,15-5)+3)

Step 3: ↓

return (5\*function\_1(5,10-5)+3)

Step 4: ↓

return (5\*function\_1(5,5-5)+3)

Step 5: ↓

return 1

Step 10: final result

218



Step 9:

5\*43+3 =218

Step 8: 

5\*8+3 =43

Step 7: 

5\*1+3 = 8

Step 6: 

return 1

|  |  |  |  |
| --- | --- | --- | --- |
| Recursion | Call | Return |  |
| 0 | function\_1(5, 15) | 5 \* function(5, 10) + 3 | 5\*43 + 3 = 218 |
| 1 | function\_1(5, 10) | 5 \* function(5, 5) + 3 | 5\*8 + 3 = 43 |
| 2 | function(5, 5) | 5 \* function(5, 0) + 3 | 5\*1 + 3 = 8 |
| 3 | function(5, 0) | 1 |  |

1. No result will be printed out.
   1. Reason: The base case of this recursive function will only happen when n(the 2nd argument) is equal to 0. However, when 12 as n(the 2nd argument), it doesn't become 0 when the recursive function is called 3rd time, instead it becomes smaller than 0. So, the condition of the base case is never met, and the recursive function will be called infinitely until memory is exceeded.

12-5 => 7-5 => 2-5 => -3

d.

int function\_1 (int m,int n){

if (n==0||n%5 != 0)

return 1;

else

return (m \* function\_1(m, n-5) + 3);

}

# 

# Page 70

1. 6
2. 1824
3. 19200

#include <iostream>

using namespace std;

int P(int);

int main()

{

cout << P(2)<<endl;

cout << P(8)<<endl;

cout << P(10);

return 0;

}

int P(int n)

{

if(n<5)

return 3\*n;

else

return P(n-2)\*4 + P(n-1)\*2;

}

# 

# Page 73

#include <iostream>

using namespace std;

int SumOfDigits (int N){

if (N/10 == 0)

return N;

else

return (N%10) + SumOfDigits((N - N%10)/10);

}

int main(){

cout << SumOfDigits(237) << endl;

cout << SumOfDigits(3518) << endl;

return 0;

}