

## Assignment 6 Functions & Iterations

### Function: getint

```
115 int getint ()
116 {
117     int invalid, num, added, endCheck;
118     char input;
119     do
120     {
121         num = 0;
122         invalid = 0;
123         added = 0;
124         endCheck = 0;
125         while (endCheck == 0 && invalid == 0)
126         {
127             input = getchar();
128             if (input >= '0' && input <= '9')
129             {
130                 num = num * 10 + (input - '0');
131                 added++;
132             }
133             else if (input == '\n' && added > 0)
134                 endCheck = 1;
135             else
136                 invalid = 1;
137         }
138         if (invalid == 1)
139             printf("[ERROR] Invalid input ! Please enter again: ");
140
141         rewind(stdin);
142     } while (invalid == 1);
143     return num;
144 }
145
```

ฟังก์ชันนี้จะรับค่าจาก input เข้ามาตรวจสอบความถูกต้องผ่านทางลูป do while ในบรรทัดที่ 119 – 143 หาก Input ที่รับมาไม่ถูกต้องจะกลับไปเริ่มลูปเพื่อรับ input ใหม่

โดยในแต่ละลูปจะมีการตรวจสอบค่า input ที่รับมาทีละตัวโดยจะแปลงจากค่า ASCII Code ของ char ทีละหลัก ไปเป็นค่าตัวเลขจำนวนเต็ม(int) ในบรรทัดที่ 127 - 136 หากถูกต้องก็จะแปลงเป็นตัวเลขตามหลักและเก็บไว้ จากนั้นตรวจสอบตัวถัดไป ในลูปบรรทัดที่ 125 -137

เมื่อทุกอย่างเสร็จสิ้นก็จะรีเทิร์นค่ากลับไปแบบตัวเลขจำนวนเต็ม

### Function: getint

```
int getint (int minInt, int maxInt)
{
    int input = getint();
    while(input < minInt || input > maxInt)
    {
        printf("[ERROR] Please enter number between %i - %i: ", minInt, maxInt);
        input = getint();
    }
    return input;
}
```

ฟังก์ชันนี้จะไปเรียกใช้งานฟังก์ชัน getint ก่อนหน้านั้น เพียงแต่ฟังก์ชันนี้จะตรวจสอบค่า input ว่าอยู่ในขอบเขตที่กำหนดหรือไม่ โดยกำหนดขอบเขตต่ำสุดและมากที่สุดผ่านทางพารามิเตอร์ โดย minInt สำหรับค่าต่ำสุด และ maxInt สำหรับค่าสูงสุด

### Function: factorial

```
158 int factorial (int n)
159 {
160     int result = 1;
161     for (int i = n; i > 0; i--)
162     {
163         result = result * i;
164     }
165     return result;
166 }
```

ฟังก์ชันคำนวณหา factorial n! โดยให้กำหนดค่า n ที่จะหาผ่านทางพารามิเตอร์ n

บรรทัดที่ 160 ประกาศตัวแปรเก็บผลลัพธ์ โดยค่าเริ่มต้นที่ 1

บรรทัดที่ 161 – 164 : ลูปคำนวณค่า factorial เช่นหา 3! ในลูปนี้ก็จะคำนวณได้ว่า  $3 \times 2 \times 1$

เมื่อจบการคำนวณก็รีเทิร์นค่าผลลัพธ์ไป

## Function: fibo

```
169 int fibo (int order)
170 {
171     if (order <= 1)
172         return order;
173     else
174         return fibo(order - 1) + fibo(order - 2);
175 }
```

ฟังก์ชันสำหรับคำนวณหา Fibonacci โดยรับพารามิเตอร์ order สำหรับลำดับฟีโบนาสี หากลำดับนั้นน้อยกว่าหรือเท่า 1 ฟังก์ชันจะรีเทิร์นค่าของลำดับกลับคืนไปเพราะไม่จำเป็นต้องคำนวณค่า แต่ถ้าหากลำดับมีค่ามากกว่า 1 ฟังก์ชันจะเรียกฟังก์ชัน fibo ของลำดับที่น้อยกว่าลำดับนี้ไปหนึ่ง และน้อยกว่าลำดับนี้ไปสอง มาบวกกันและรีเทิร์นค่ากลับมา เช่น ต้องการหาลำดับที่ 5 หรือก็คือ fibo(5) จะได้ค่า fibo(4) + fibo(3) ซึ่ง fibo(4) ก็จะได้ค่าเท่ากับ fibo(3) + fibo(2) นั่นเอง

สรุปก็คือหาลำดับที่ต้องการมากกว่า 1 ฟังก์ชันนี้จะเรียกฟังก์ชันหาลำดับที่น้อยกว่าให้หนึ่งลำดับ และน้อยกว่าสองลำดับมากบวกกัน ซึ่งฟังก์ชันทั้งสองที่เรียกมาก็จะซ้ำเรียกฟังก์ชันหาลำดับที่มากกว่า 1 เมื่อลำดับคือ 0 ค่าของฟังก์ชันคือ 0 และเมื่อลำดับคือ 1 ค่าของฟังก์ชันคือ 1 และก็จะรีเทิร์นค่ากลับไปให้ลำดับที่สูงกว่าที่เรียกฟังก์ชันไปคำนวณ

**ตัวอย่าง:** เรียกฟังก์ชัน fibo(4)

ลำดับที่	เงื่อนไข	ค่าของฟังก์ชัน
4	$n > 1 : \text{fibo}(n - 1) + \text{fibo}(n - 2)$	$\text{fibo}(3) + \text{fibo}(2)$
3	$n > 1 : \text{fibo}(n - 1) + \text{fibo}(n - 2)$	$\text{fibo}(2) + \text{fibo}(1)$
2	$n > 1 : \text{fibo}(n - 1) + \text{fibo}(n - 2)$	$\text{fibo}(1) + \text{fibo}(0)$
1	$n \leq 1 : n$	1
0	$n \leq 1 : n$	0

ผลลัพธ์ที่ได้คือ  $0 + 1 + 1 + 1 = 3$

เมื่อฟังก์ชันหาค่าได้แล้วก็จะรีเทิร์นค่ากลับไป

## Function: quickSum

```
188 double quickSum(int start, int n, double d)
189 {
190     double result = 0;
191     for (int i = start; i <= n; i++)
192     {
193         result += i * d;
194     }
195     return result;
196 }
```

ฟังก์ชันนี้จะหาผลรวมของลำดับอนุกรม โดยรับพารามิเตอร์ start สำหรับลำดับแรกที่เริ่มคำนวณ, พารามิเตอร์ n สำหรับลำดับสุดท้ายที่คำนวณ, พารามิเตอร์ d ค่าการเพิ่มขึ้นของลำดับ

บรรทัดที่ 190: ประกาศตัวแปร result ไว้สำหรับเก็บผลลัพธ์ โดยให้ค่าเริ่มต้นคือ 0

บรรทัดที่ 191 – 194: เริ่มลูปการคำนวณโดยเริ่มตั้งแต่ลำดับ start ถึงลำดับที่ n โดย i คือค่าลำดับในลูปนั้น

และบวกเพิ่มค่าของลำดับในลูป ( $i * d$ ) ไว้กับ result

ตัวอย่าง: start = 1, n = 3, d = 2

รอบที่	i	$i * d$	result
1	1	2	2
2	2	4	6
3	3	6	12

ผลลัพธ์ที่ได้คือ 12

เมื่อฟังก์ชันคำนวณค่าผลลัพธ์เสร็จแล้วก็จะรีเทิร์นค่าผลลัพธ์กลับไป

## Function: printHeader

```
66 void printHeader(char text[], int length)
67 {
68     int starCount = length; // star
69     int amount = strlen(text),
70     lineAmount = 1;
71     for (int i = 1; i <= length; i++)
72         printf("*");
73     printf("\n");
74
75     // Get line amount
76     for (int i = 0; i < amount; i++)
77     {
78         if (text[i] == '\n')
79             lineAmount += 1;
80     }
81     // Start line
82     for (int line = 1; line <= lineAmount; line++)
83     {
84         int currentLine = 1, lineTextAmount = 0, fistTextIndex = 0;
85         for (int i = 0; i < amount && currentLine <= line; i++)
86         {
87             if (text[i] == '\n')
88                 currentLine++;
89             else if (text[i] != '\n' && currentLine == line)
90             {
91                 if (lineTextAmount == 0)
92                     fistTextIndex = i;
93                 if (lineTextAmount <= length - 3)
94                     lineTextAmount++;
95             }
96         }
97
98         int startAt = (length/2) - lineTextAmount/2;
99         int textIndex = 0 + fistTextIndex;
100         for (int i = 1; i <= length; i++)
101         {
102             if (i == 1 || i == length - 1)
103                 printf("*");
104             if (i==length)
105                 printf("\n");
106             else if (i < startAt || i > (startAt - 1) + lineTextAmount)
107                 printf(" ");
108             else if (text[textIndex] == '\n') {
109                 printf(" ");
110             }
111             else printf("%c", text[textIndex++]);
112         }
113     }
114     for (int i = 1; i <= length; i++)
115         printf("*");
116     printf("\n");
117 }
```

ฟังก์ชันสำหรับใช้ปรี้นแสดงผลสำหรับส่วนหัวข้อต่างๆ สามารถกำหนดความยาวของกรอบส่วนหัวได้ โดยฟังก์ชันจะรับข้อความผ่านทางพารามิเตอร์ text และ length สำหรับความยาวของกรอบ จากนั้นตัวฟังก์ชันก็คำนวณหาระยะเว้นวรรคเพื่อจัดข้อความให้กึ่งกลาง โดยสูตรการคำนวณหาจำนวนการเว้นวรรคคือ  $(\text{ความยาวกรอบที่ตัดส่วนขอบออกไป})/2 - (\text{จำนวนตัวอักษรในบรรทัด})/2$  พร้อมทั้งแสดงผลข้อความให้อย่างสวยงาม

### ตัวอย่าง

โค้ด: printHeader("Hello world", 50);

ผลลัพธ์:

```
*****
*               Hello world !               *
*****
```

## Function: test1

```
207 void test1 (int n)
208 {
209     printHeader("Test1\n", 30);
210     printf("* i * fi * sum *\n");
211     printf("*****\n");
212     double sum = 0;
213     for (int i = 1; i <= n; i++)
214     {
215         double fi = quickSum(1, i, 2.0) / factorial(i);
216         sum += fi;
217         printf("* %2d * %9lf * %9lf *\n", i, fi, sum);
218     }
219     printf("*****\n");
220     printf("ans = %f \n", sum);
221 }
```

ฟังก์ชัน test 1 ใช้สำหรับการหาผลรวมจาก n เทอมของ

$$\frac{2}{1} + \frac{2+4}{1*2} + \frac{2+4+6}{1*2*3} + \frac{2+4+6+8}{1*2*3*4} + \dots$$

บรรทัดที่ 209 – 211: เป็นส่วนของหัวข้อ และหัวตารางสำหรับตกแต่ง

บรรทัดที่ 212: ประกาศตัวแปร sum ใช้สำหรับเก็บผลรวม

บรรทัดที่ 213 – 218: คำนวณผลลัพธ์ในแต่ละเทอมจากเทอมที่ 1 ถึงเทอมที่ n

โดยใช้ฟังก์ชัน quickSum สำหรับหาผลรวมของ 2 + 4 + 6 + ...

และใช้ฟังก์ชัน factorial สำหรับหา 1 \* 2 \* ...

จากนั้นนำผลลัพธ์ไปรวมกับ sum พร้อมทั้งปρί้นตารางแสดงค่าของเทอมในรอบนั้น

บรรทัดที่ 219 – 220: หลังจากหาค่าเสร็จแล้วก็ปρί้นปิดตารางเพื่อความสวยงาม และปρί้นผลรวมที่ได้จากการทำ

คำนวณ ในโจทย์ข้อที่ 1 กำหนดให้หา 10 เทอม

โค้ด: test1(10)

ผลลัพธ์:

```
*****
*                Test1                *
*                *                      *
*****
* i * fi * sum *
*****
* 1 * 2.000000 * 2.000000 *
* 2 * 3.000000 * 5.000000 *
* 3 * 2.000000 * 7.000000 *
* 4 * 0.833333 * 7.833333 *
* 5 * 0.250000 * 8.083333 *
* 6 * 0.058333 * 8.141667 *
* 7 * 0.011111 * 8.152778 *
* 8 * 0.001786 * 8.154563 *
* 9 * 0.000248 * 8.154812 *
* 10 * 0.000030 * 8.154842 *
*****
ans = 8.154842
```

## Function: test2

```
223 void test2 (int n)
224 {
225     printHeader("\nTest 2\n", 24);
226     printf("*count* i * fi *\n");
227     printf("*****\n");
228
229     int i = 1;
230     int add = 0;
231     while (add < n)
232     {
233         int fi = fibo(i);
234         if (fi % 2 == 0)
235         {
236             printf("* %3d * %4d * %7d *\n", ++add, i, fi);
237         }
238         i++;
239     }
240     printf("*****\n");
241 }
```

ฟังก์ชันสำหรับหาค่า Fibonacci ที่เป็นเลขคู่ โดยมีพารามิเตอร์ n สำหรับกำหนดจำนวนเทอม

บรรทัดที่ 225 – 227: ปริ้นส่วนหัวเพื่อตกแต่ง

บรรทัดที่ 229 – 230: ประกาศตัวแปร i ใช้สำหรับนับลำดับ และประกาศตัวแปร add เพื่อใช้นับจำนวนที่เจอที่หารด้วย 2 ลงตัว

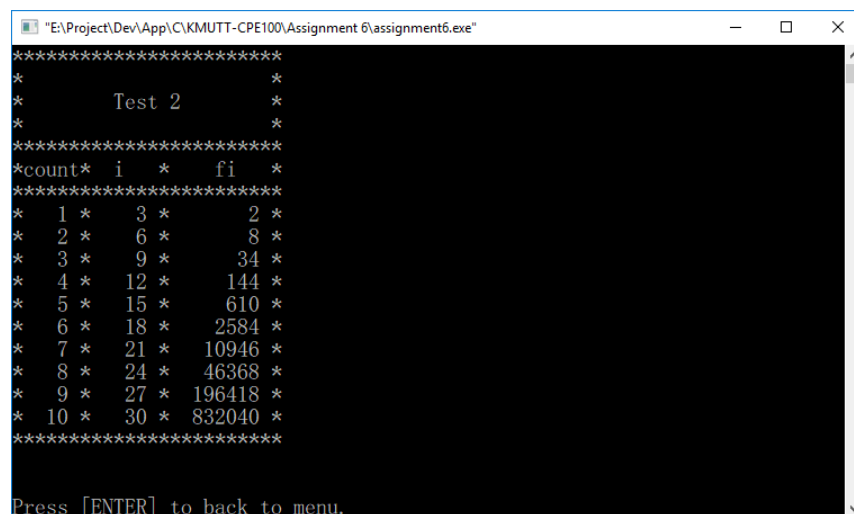
บรรทัดที่ 231 – 339: ลูปหาค่าที่หารด้วยสองลงตัว จนกว่าค่าที่หาเจอจะครบเทอม

บรรทัดที่ 233: โดยประกาศค่า fi สำหรับเก็บค่าของ fibonacci ในลำดับที่ i โดยใช้ฟังก์ชัน fibo

บรรทัดที่ 234: ตรวจสอบว่าค่า fi หารด้วยสองลงตัวไหม หากหารลงตัวก็ทำจากปρί้นลงในตาราง และนับค่า add เพิ่ม

จากโจทย์ข้อที่ 2 ให้เราหา 10 นำมาเขียนเป็นโค้ด คือ test2(10);

ผลลัพธ์:



```
"E:\Project\Dev\App\C\KMUTT-CPE100\Assignment 6\assignment6.exe"
*****
*                               *
*           Test 2              *
*                               *
*****
*count* i * fi *
*****
* 1 * 3 * 2 *
* 2 * 6 * 8 *
* 3 * 9 * 34 *
* 4 * 12 * 144 *
* 5 * 15 * 610 *
* 6 * 18 * 2584 *
* 7 * 21 * 10946 *
* 8 * 24 * 46368 *
* 9 * 27 * 196418 *
* 10 * 30 * 832040 *
*****
Press [ENTER] to back to menu.
```

### Function: test3

```
243 void test3 (int a)
244 {
245     printHeader("\nTest 3\n", 21);
246     printf("* no * fi * sum *\n");
247     printf("*****\n");
248     int sum = 0;
249     int i = 1;
250     while (sum < a)
251     {
252         int fi = (1 + ((i - 1) * 2)) * (40 - (2 * (i - 1)));
253         sum += fi;
254         printf("* %2d * %4d * %5d *\n", i, fi, sum);
255         i++;
256     }
257     printf("*****\n");
258     printf("n = %d, ans = %d\n", --i, sum);
259 }
```

ฟังก์ชันหาจำนวนเทอมและผลรวมต่ำสุดของอนุกรม

**$(1 \times 40) + (3 \times 38) + (5 \times 36) + \dots + (39 \times 2)$**  ที่มีค่าเกิน a (กำหนดผ่านพารามิเตอร์ a)

บรรทัดที่ 245 - 247: ปรี้นส่วนหัวเพื่อตกแต่ง

บรรทัดที่ 248 - 249: ประกาศตัวแปร sum สำหรับเก็บผลรวม และ i สำหรับนับจำนวนเทอม

บรรทัดที่ 250 - 257: เริ่มลูปโดยทำไปเรื่อยๆ หากค่า sum ยังน้อยกว่าค่า a

บรรทัดที่ 252 - 255: นำเทอม  **$(1 \times 40) + (3 \times 38) + (5 \times 36) + \dots + (39 \times 2)$**

มาเขียนในรูปแบบของโค้ด และบวกเพิ่มเข้าไปใน sum พร้อมทั้งปรี้นแสดงผลตาราง

และนับค่า i เพิ่มขึ้น

บรรทัดที่ 257 - 258: ปรี้นปิดตารางและปรี้นแสดงจำนวนเทอมและผลรวมต่ำสุดของอนุกรมที่เกินค่า a

จากโจทย์ข้อที่ 3 ให้หาค่าที่เกิน 5000

เขียนเป็นโค้ด คือ test3(5000);

ผลลัพธ์:

```
"E:\Project\Dev\App\C\KMUTT-CPE100\Assignment 6\assignment6.exe"
*****
*
*      Test 3      *
*
*****
* no * fi * sum *
*****
* 1 * 40 * 40 *
* 2 * 114 * 154 *
* 3 * 180 * 334 *
* 4 * 238 * 572 *
* 5 * 288 * 860 *
* 6 * 330 * 1190 *
* 7 * 364 * 1554 *
* 8 * 390 * 1944 *
* 9 * 408 * 2352 *
* 10 * 418 * 2770 *
* 11 * 420 * 3190 *
* 12 * 414 * 3604 *
* 13 * 400 * 4004 *
* 14 * 378 * 4382 *
* 15 * 348 * 4730 *
* 16 * 310 * 5040 *
*****
n = 16, ans = 5040
```



## Function: test4

```
261 void test4 (int a, int b, int c)
262 {
263     printHeader("\nTest 4\n", 21);
264     printf("* no * i * sum *\n");
265     printf("*****\n");
266     int n = 1, sum = 0;
267     for (int i = a; i<=b; i++)
268     {
269         if (i%c==0)
270         {
271             sum += i;
272             printf("* %2d * %4d * %4d *\n", n++, i, sum);
273         }
274     }
275     printf("*****\n");
276     printf("count = %d, ans = %d\n", (n-1), sum);
277 }
```

ฟังก์ชันหาผลบวกและจำนวนเทอมของจำนวนเต็มที่อยู่ระหว่าง a ถึง b และหารด้วย c ลงตัว (a, b, c กำหนดผ่านทางพารามิเตอร์ โดยที่ a < b)

บรรทัดที่ 263 – 265: ปรับส่วนหัวตกแต่ง

บรรทัดที่ 266: ประกาศค่า n เพื่อเก็บจำนวนเทอม และค่า sum สำหรับผลบวก

บรรทัดที่ 267 – 274: ลูปคำนวณค่าตั้งแต่ค่า a ถึง b

บรรทัดที่ 269 – 273: หากค่าระหว่าง a ถึง b หารด้วย c ลงตัว บวกเพิ่มไปใน sum นับ n เพิ่มขึ้น และปริ้นแสดงลงตาราง

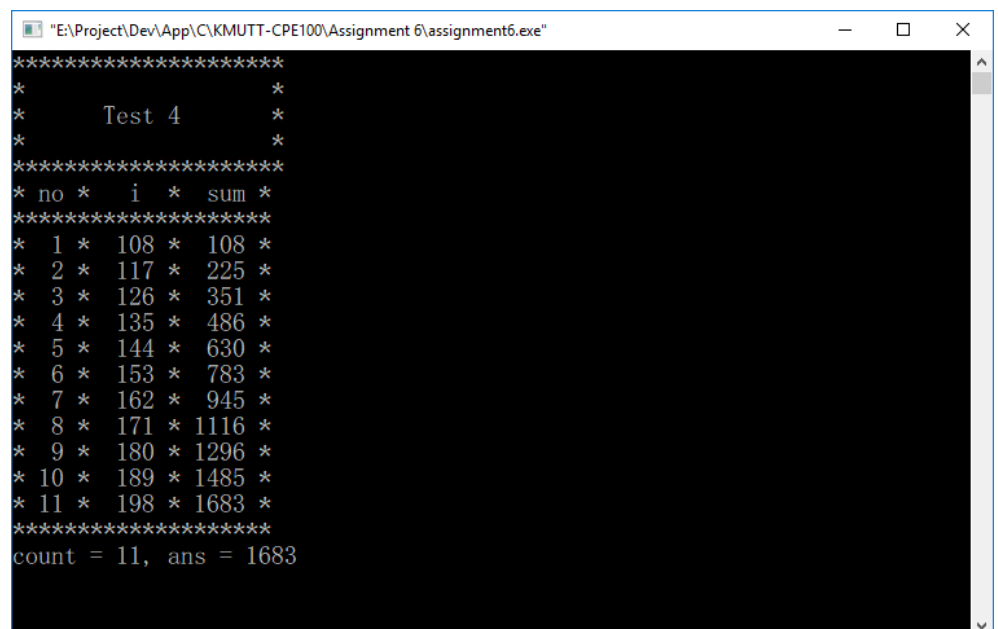
บรรทัดที่ 275 – 276: ปริ้นปิดตารางและปริ้นสรุปจำนวนเทอมและผลรวม

จากโจทย์ข้อที่ 4 ให้หาผลรวมระหว่าง 100 – 200 ที่หารด้วย 9 ลงตัว

โค้ด:

test4(100, 200, 9)

ผลลัพธ์:



```
*****
*          *
*   Test 4   *
*          *
*****
* no *   i *  sum *
*****
* 1 * 108 * 108 *
* 2 * 117 * 225 *
* 3 * 126 * 351 *
* 4 * 135 * 486 *
* 5 * 144 * 630 *
* 6 * 153 * 783 *
* 7 * 162 * 945 *
* 8 * 171 * 1116 *
* 9 * 180 * 1296 *
* 10 * 189 * 1485 *
* 11 * 198 * 1683 *
*****
count = 11, ans = 1683
```

## Function: test5

```
279 void test5 (int n)
280 {
281     printHeader("\nTest 5\n", 13);
282     printf("* no * term *\n");
283     printf("*****\n");
284
285     int a,b, order = 1;
286     for (int i = 0; i < n; i++)
287     {
288         a = 1 + (i * 3);
289         for (int j = 0; j < n; j++)
290         {
291             b = 1 + (j * 5);
292             if (a == b)
293             {
294                 printf("* %2d * %4d *\n", order, a);
295                 j = n;
296                 order++;
297             }
298         }
299     }
300     printf("*****\n");
301 }
```

ฟังก์ชันสำหรับแสดงและนับจำนวนพจน์ที่ซ้ำกันใน n พจน์แรก(กำหนดในพารามิเตอร์) ของลำดับเลขคณิต 1, 4, 7, 10, 13, 16, .. และ 1, 6, 11, 16, 21, 26, 31

บรรทัดที่ 281 – 284: ปรับส่วนหัวเพื่อตกแต่ง

บรรทัดที่ 285: ประกาศตัวแปร a, b สำหรับแทนลำดับเลขคณิต และ order สำหรับนับลำดับ

บรรทัดที่ 286 – 299: ลูปในลำดับเลขคณิตของ a

    บรรทัดที่ 288: เก็บค่าลำดับ a

    บรรทัดที่ 289: ลูปในลำดับเลขคณิตของ b

        บรรทัดที่ 291: เก็บค่าลำดับ b

        บรรทัดที่ 292: ตรวจสอบว่าค่า a และ b เท่ากันหรือไม่

            บรรทัดที่ 294 -296: หากเท่ากันให้ปรับแสดงลงในตาราง และหยุดลูป b เพื่อให้เริ่มลูป a ใหม่ และนับ order เพิ่มขึ้น

บรรทัดที่ 300: ปรับปิดตาราง

จากโจทย์ข้อที่ 5 ให้พจน์ 100 พจน์แรกที่ซ้ำกันในลำดับเลขคณิตทั้งสอง

เขียนโค้ดได้ว่า test5(100);

ผลลัพธ์:

```
"E:\Project\Dev\App\C\KMUTT-CPE100\Assignment 6\assignment6.exe"
*****
*                               *
*   Test 5   *
*                               *
*****
* no * term *
*****
* 1 *   1 *
* 2 *  16 *
* 3 *  31 *
* 4 *  46 *
* 5 *  61 *
* 6 *  76 *
* 7 *  91 *
* 8 * 106 *
* 9 * 121 *
*10 * 136 *
*11 * 151 *
*12 * 166 *
*13 * 181 *
*14 * 196 *
*15 * 211 *
*16 * 226 *
*17 * 241 *
*18 * 256 *
*19 * 271 *
*20 * 286 *
*****

Press [ENTER] to back to menu. _
```