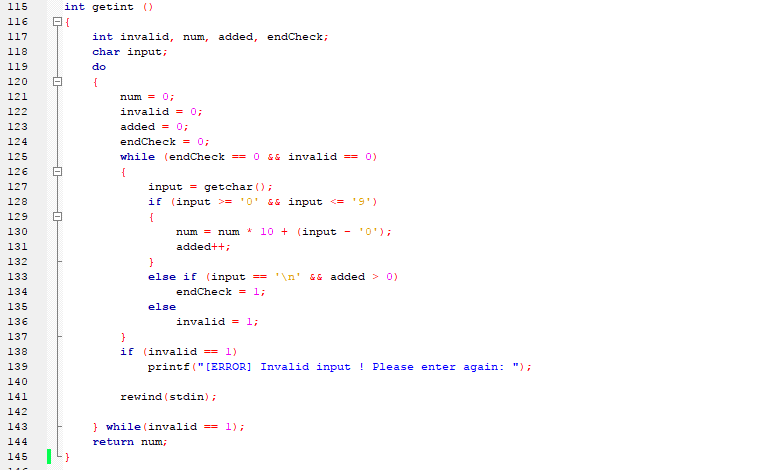
**Assigment 6 Functions & Iterations**

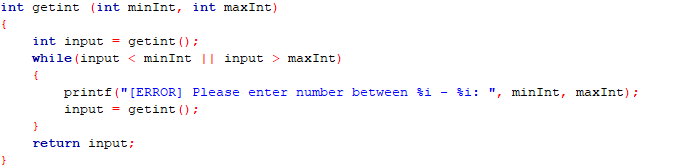
**Function: getint**

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

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

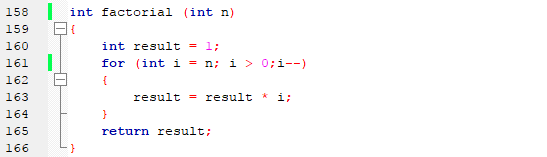
เมื่อทุกอย่างเสร็จสิ้นก็จะรีเทิร์นค่ากับไปแบบตัวเลขจำนวนเต็ม

**Function: getint**



ฟังก์ชันนี้จะไปเรียกใช้งานฟังก์ชัน getint ก่อนหน้านี้ เพียงแต่ฟังก์ชันนี้จะตรวจสอบค่า input ว่าอยู่ในขอบเขตที่กำหนดหรือไม่ โดยกำหนดขอบเขตต่ำสุดและมากสุดผ่านทางพารามิเตอร์ โดย minInt สำหรับค่าต่ำสุด และ maxInt สำหรับค่าสูงสุด

**Function: factorial**



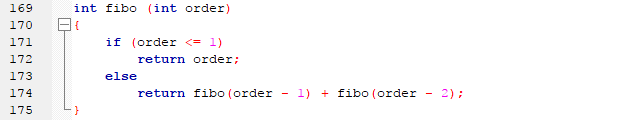
ฟังก์ชันคำนวณหา factorial n! โดยให้กำหนดค่า n ที่จะหาผ่านทางพารามิเตอร์ n

บรรทัดที่ 160 ประกาศตัวแปรเก็บผลลัพธ์ โดยค่าเริ่มต้นที่ 1

บรรทัดที่ 161 – 164 : ลูปคำนวณค่า factorial เช่นหา 3! ในลูปนี้ก็จะคำนวณได้ว่า 3 x 2 x 1

เมื่อจบการคำนวณก็รีเทิร์นค่าผลลัพธ์ไป

**Function: fibo**



ฟังก์ชันสำหรับคำนวณหา Fibonacci โดยรับพารามิเตอร์ order สำหรับลำดับฟิโบนาชี หากลำดับนั้นน้อยกว่าหรือเท่า 1 ฟังก์ชันจะรีเทิร์นค่าของลำดับกลับคืนไปเพราะไม่จำเป็นต้องคำนวณหาค่า 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) นั้นเอง

สรุปก็คือหากลำดับที่ต้องหามากกว่า 1 ฟังก์ชันนี้ก็จะเรียกฟังก์ชันหาค่าลำดับที่น้อยกว่าให้หนึ่งลำดับ และน้อยกว่าสองลำดับมากบวกกัน ซึ่งฟังก์ชันทั้งสองที่เรียกมาก็จะทำซ้ำเรียกฟังก์ชันหาค่ามาอีกทีหากลำดับยังมากกว่า 1 เมื่อลำดับคือ 0 ค่าของฟังก์ชันคือ 0 และเมื่อลำดับคือ 1 ค่าของฟังก์ชันคือ 1 และก็รีเทิร์นค่ากลับไปให้ลำดับที่สูงกว่าที่เรียกฟังก์ชันไปคำนวณ

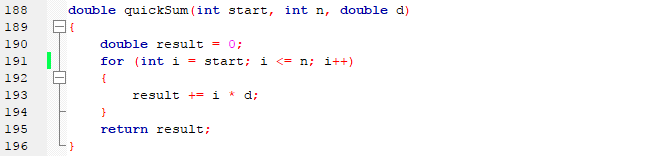
**ตัวอย่าง:** เรียกฟังก์ชัน fibo(4)

|  |  |  |
| --- | --- | --- |
| ลำดับที่ | เงื่อนไข | ค่าของฟังก์ชัน |
| 4 | n > 1 : fibo(n - 1) + fibo(n - 2) | fibo(3) + fibo(2) |
| 3 | n > 1 : fibo(n - 1) + fibo(n - 2) | fibo(2) + fibo(1) |
| 2 | n > 1 : fibo(n - 1) + fibo(n - 2) | fibo(1) + fibo(0) |
| 1 | n <= 1 : n | 1 |
| 0 | n <= 1 : n | 0 |

ผลลัพธ์ที่ได้คือ 0 + 1 + 1 + 1 = 3

เมื่อฟังก์ชันหาค่าได้แล้วก็จะรีเทิร์นค่ากลับไป

**Function: quickSum**



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

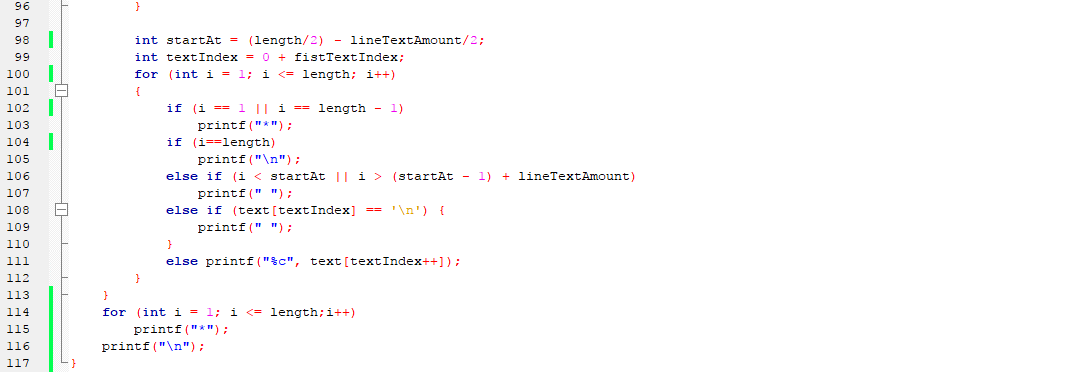
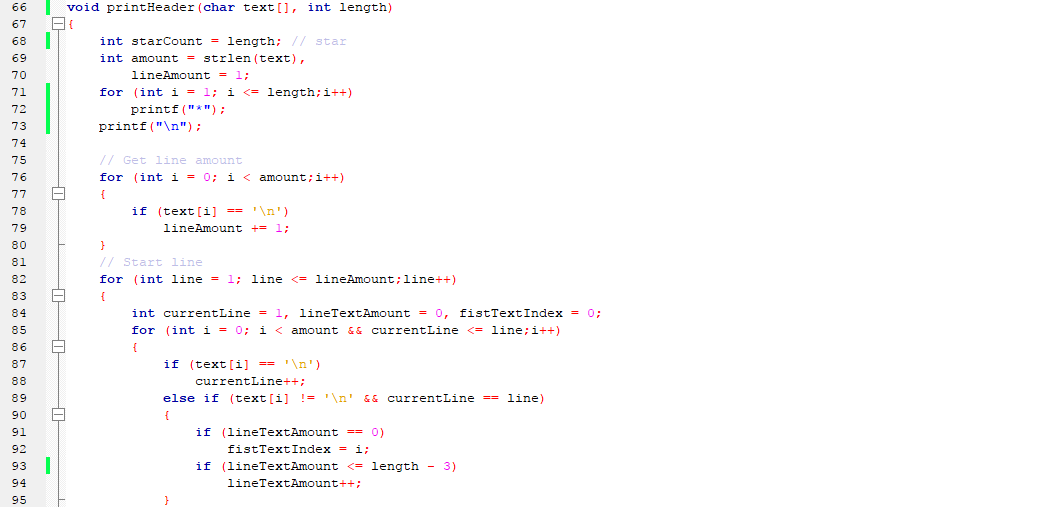
ตัวอย่าง: start = 1, n = 3, d = 2

|  |  |  |  |
| --- | --- | --- | --- |
| รอบที่ | i | i \* d | result |
| 1 | 1 | 2 | 2 |
| 2 | 2 | 4 | 6 |
| 3 | 3 | 6 | 12 |

ผลลัพธ์ที่ได้คือ 12

เมื่อฟังก์ชันคำนวณค่าผลลัพธ์เสร็จแล้วก็จะรีเทิรน์ค่าผลลัพธ์กลับไป

**Function: printHeader**

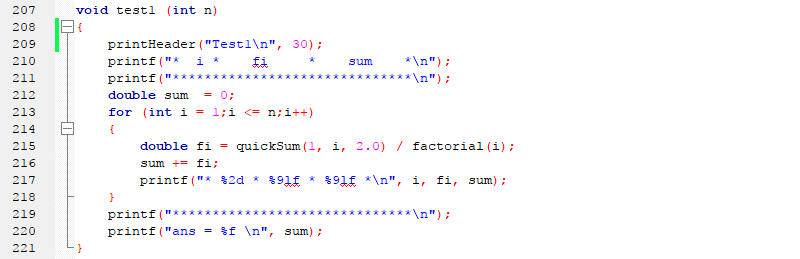


ฟังก์ชันสำหรับใช้ปริ้นแสดงผลสำหรับส่วนหัวข้อต่างๆ สามารถกำหนดความยาวของกรอบส่วนหัวได้ โดยฟังก์ชันจะรับข้อความผ่านทางพารามิเตอร์ text และ length สำหรับความยาวของกรอบ จากนั้นตัวฟังก์ชันก็คำนวณหาระยะเว้นวรรคเพื่อจัดข้อความให้กึ่งกลาง โดยสูตรการคำนวณหาจำนวนการเว้นวรรคคือ   
(ความยาวกรอบที่ตัดส่วนขอบออกไป)/2 – (จำนวนตัวอักษรในบรรทัด)/2   
พร้อมทั้งแสดงผลข้อความให้อย่างสวยงาม

**ตัวอย่าง**โค้ด: printHeader(“Hello world”, 50);

ผลลัพธ์: 

**Function: test1**



ฟังก์ชัน test 1 ใช้สำหรับการหาผลรวมจาก n เทอมของ



บรรทัดที่ 209 – 211: เป็นส่วนของหัวข้อ และหัวตารางสำหรับตกแต่ง

บรรทัดที่ 212: ประกาศตัวแปร sum ใช้สำหรับเก็บผลรวม

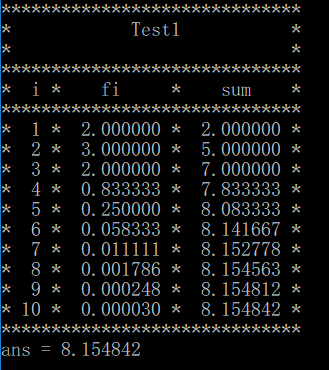
บรรทัดที่ 213 – 218: คำนวณผลลัพธ์ในแต่ละเทอมจากเทอมที่ 1 ถึงเทมอที่ n

โดยใช้ฟังก์ชัน quickSum สำหรับหาผลรวมของ 2 + 4 + 6 + …

และใช้ฟังก์ชัน factorial สำหรับหา 1 \* 2 \* …

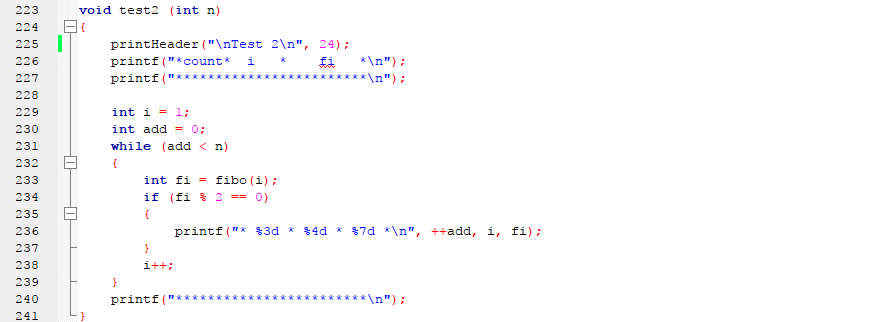
จากนั้นนำผลลัพธ์ไปรวมกับ sum พร้อมทั้งปริ้นตารางแสดงค่าของเทอมในรอบนั้น

บรรทัดที่ 219 – 220: หลังจากหาค่าเสร็จแล้วก็ปริ้นปิดตารางเพื่อความสวยงาม และปริ้นผลรวมที่ได้จากการทำคำนวณ **ในโจทย์ข้อที่ 1 กำหนดให้หา 10 เทอม**

**โค้ด:** test1(10)

**ผลลัพธ์:**

**Function: test2**

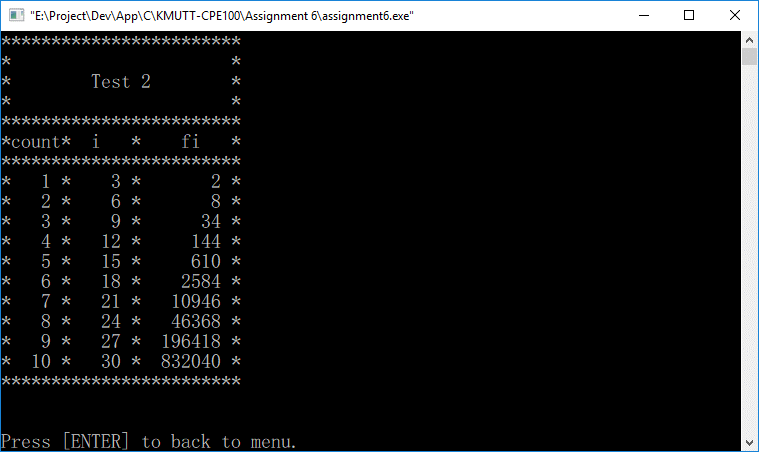


ฟังก์ชันสำหรับหาค่า Fibonacci ที่เป็นเลขคู่ โดยมีพารามิเตอร์ n สำหรับกำหนดจำนวนเทอม

บรรทัดที่ 225 – 227: ปริ้นส่วนหัวเพื่อตกแต่ง

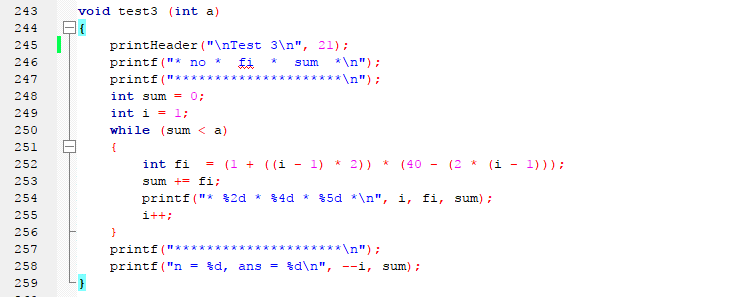
บรรทัดที่ 229 – 230: ประกาศตัวแปร i ใช้สำหรับนับลำดับ และประกาศตัวแปร add เพื่อใช้นับจำนวนที่เจอที่หารด้วย 2 ลงตัว

บรรทัดที่ 231 – 339: ลูปหาค่าที่หารด้วยสองลงตัว จนกว่าค่าที่หาเจอจะครบเทอม   
 บรรทัดที่ 233: โดยประกาศค่า fi สำหรับเก็บค่าของ fibonacci ในลำดับที่ i โดยใช้ฟังก์ชัน fibo  
 บรรทัดที่ 234: ตรวสอบว่าค่า fi หารด้วยสองลงตัวไหม หากหารลงตัวก็ทำจากปริ้นลงในตาราง   
 และนับค่า add เพิ่ม

จากโจทย์ข้อที่ 2 ให้เราหา 10 นำมาเขียนเป็นโค้ด คือ test2(10);

**ผลลัพธ์:**

**Function: test3**



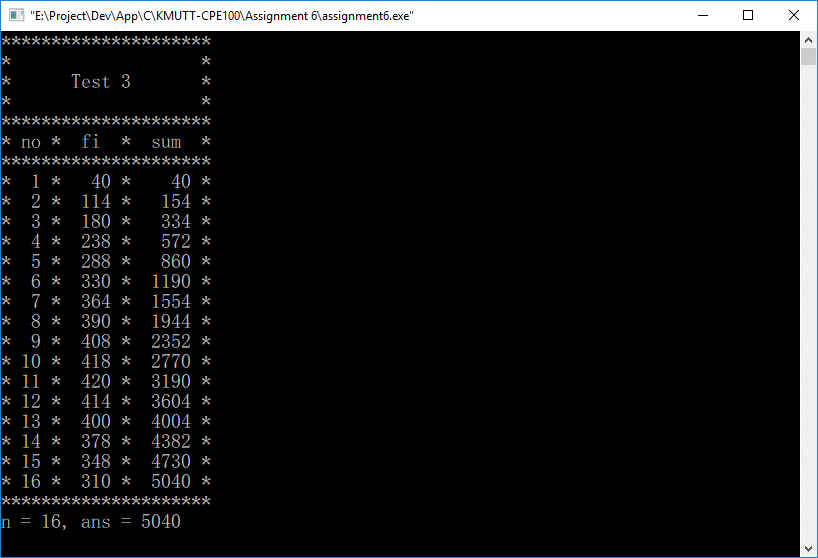
ฟังก์ชันหาจำนวนเทอมและผลรวมต่ำสุดของอนุกรม  ที่มีค่าเกิน a (กำหนดผ่านพารามิเตอร์ a)

บรรทัดที่ 245 – 247: ปริ้นส่วนหัวเพื่อตกแต่ง

บรรทัดที่ 248 – 249: ประกาศตัวแปร sum สำหรับเก็บผลรวม และ i สำหรับนับจำนวนเทอม

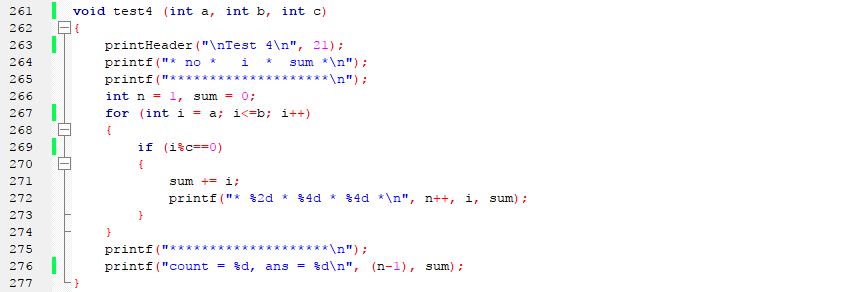
บรรทัดที่ 250 – 257: เริ่มลูปโดยทำไปเรื่อยๆ หากค่า sum ยังน้อยกว่าค่า a  
 บรรทัดที่ 252 - 255: นำเทอม   
 มาเขียนในรูปแบบของโค้ด และบวกเพิ่มเข้าไปใน sum พร้อมทั้งปริ้นแสดงผลลงตาราง

และนับค่า i เพิ่มขึ้น

บรรทัดที่ 257 -258: ปริ้นปิดตารางและปริ้นแสดงจำนวนเทอมและผลรวมต่ำสุดของอนุกรมที่เกินค่า a

จากโจทย์ข้อที่ 3 ให้หาค่าที่เกิน 5000   
**เขียนเป็นโค้ด คือ test3(5000);  
ผลลัพธ์:**

**Function: test4**



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

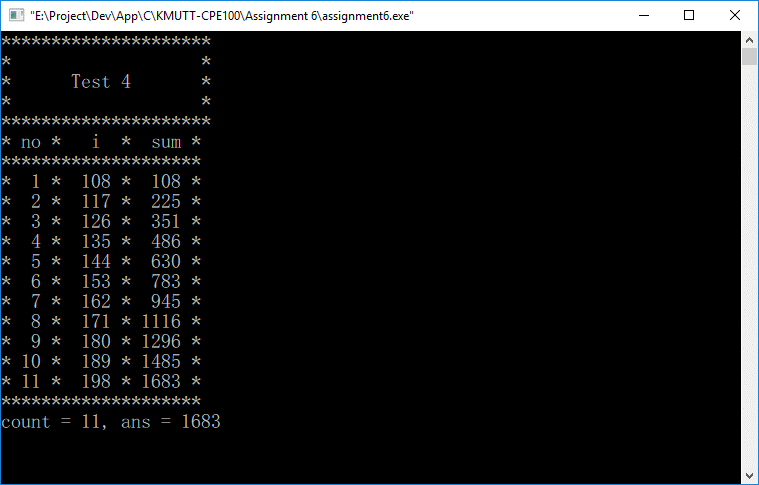
บรรทัดที่ 263 – 265: ปริ้นส่วนหัวตกแต่ง

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

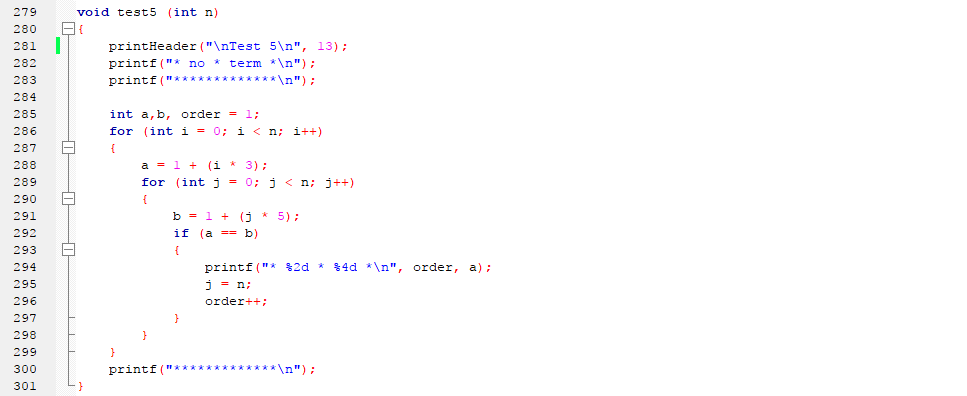
บรรทัดที่ 275 – 276: ปริ้นปิดตารางและปริ้นสรุปจำนวนเทอมและผลรวม

****จากโจทย์ข้อที่ 4 ให้หาผลรวมระหว่าง 100 – 200 ที่หารด้วย 9 ลงตัว

**โค้ด:**   
test4(100, 200, 9)

**ผลลัพธ์:**

**Function: test5**



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

บรรทัดที่ 281 – 284: ปริ้นส่วนหัวเพื่อตกแต่ง

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: หากเท่ากันให้ปริ้นแสดงลงในตาราง และหยุดลูป b เพื่อให้เริ่มลูป  
 a ใหม่ และนับ order เพิ่มขึ้น

บรรทัดที่ 300: ปริ้นปิดตาราง

จากโจทย์ข้อที่ 5 ให้พจน์ 100 พจน์แรกที่ซ้ำกันในลำดับเลขคณิตทั้งสอง

เขียนโค้ดได้ว่า test5(100);

**ผลลัพธ์**: