CP 111 Computer Programming, 2/2554, Section B01/B02

Programming Homework 05

กำหนดส่งภายในวันจันทร์ที่ 23 ม.ค. 2555 เวลา 23.59 น.

คำเตือน

นิสิตสามารถจับกลุ่มปรึกษาหารือกันได้ แต่นิสิตแต่ละคนในกลุ่มจะต้องลงมือเขียนโปรแกรมด้วยตนเองและส่งโปรแกรมของตนเองโดยไม่ขึ้นอยู่กับสมาชิกในกลุ่ม ***และอย่านำโปรแกรมของตนเองให้กับผู้อื่นดู***

ถ้าตรวจพบว่านิสิตนำงานของคนอื่นมาส่งหรือนิสิตก็อปปี้ แก้ไข ดัดแปลงโปรแกรมของผู้อื่น จะมีโทษคือปรับตก (ได้เกรด E) ในวิชานี้ทันที

หมายเหตุ

1. ข้อกำหนดการตั้งชื่อไฟล์การบ้านคือ xxx_**HW**yy_**Q**zz.* โดยที่ xxx = รหัสสามตัวท้ายของเลขประจำตัว, yy = หมายเลข homework, zz = หมายเลขข้อในแต่ละ homework (ตัวอย่างเช่น ไฟล์โปรแกรม ข้อ 1 การบ้านครั้งที่ 1 ของนิสิตรหัส 000 คือ 000_HW01_Q01.c) และในแต่ละข้อ

2. การส่งการบ้าน ให้ส่งไฟล์โปรแกรมไปตรวจกับระบบ Café-grader ที่ http://cslab.swu.ac.th/grader

3. การเขียนโปรแกรมที่คำนวณกับตัวเลขที่มีจุดทศนิยม ให้ประกาศตัวแปรที่เกี่ยวข้องเป็นแบบ `double` (Double-precision floating point)

1. Problem name: `Cal_Integration_simpson`

จงเขียนโปรแกรมเพื่อหาค่า $\int_a^b (k_1x^2 + k_2x + k_3)dx$ โดยที่ k_1, k_2, k_3, a และ b เป็นค่าแบบ `double` จะถูกป้อนมาทางคีย์บอร์ด เรียงต่อกัน โดยกำหนดให้การคำนวณ numerical integration $\int_a^b f(x)\,dx$ ทำโดยใช้สูตร simpson rule ดังสมการข้างล่าง

$$\int_a^b f(x)dx = \left[\frac{1}{3}*c*f(x_0) + \frac{4}{3}*c*f(x_1) + \frac{2}{3}*c*f(x_2) + \frac{4}{3}*c*f(x_3) + ... + \frac{2}{3}*c*f(x_{n-2}) + \frac{4}{3}*c*f(x_{n-1}) + \frac{1}{3}*c*f(x_n)\right]$$

โดยที่ $x_i = a + (c*i)\ where\ i = 0,1,2,...,n$ และ $c = \frac{b-a}{n}$

<u>หมายเหตุ</u>

(1) สูตรข้างบนนี้เป็นสูตรการหาค่า integration ที่มีการใช้กันอย่างแพร่หลายและมีความแม่นยำกว่าวิธีการที่ได้เขียนโปรแกรมไปก่อนหน้านี้

(2) สูตรข้างบนนี้จะหาผลรวมตั้งแต่เทอม i=0 (หรือ 1/3*c*f(x_0)) จนถึงเทอม n (หรือ 1/3*c*f(x_n)) ทั้งหมด n+1 เทอม

(3) ค่าสัมประสิทธิ์ที่คูณกับ f(x_0) และ f(x_n) ในเทอมที่ 0 และ n คือ 1/3

(4) ค่าสัมประสิทธิ์ในเทอมที่ 1, 3, 5, .., n-1 คือ 4/3

(5) ค่าสัมประสิทธิ์ในเทอมที่ 2, 4, 6, .., n-2 คือ 2/3

กำหนดให้เขียนโปรแกรมคำนวณค่า integration จากผลรวมตามสูตรข้างบนโดยที่ **n=100 และแสดงผลลัพธ์ออกมาเป็นเลขทศนิยม 12 หลัก** นอกจากนี้ให้แสดงค่าของ integration ที่คำนวณได้จากวิธี analytic ตามสูตรข้างล่าง ในบรรทัดถัดไป

$$\int_a^b (k_1x^2 + k_2x + k_3)dx = \left.\frac{k_1x^3}{3} + \frac{k_2x^2}{2} + k_3x\right|_a^b = \left(\frac{k_1b^3}{3} + \frac{k_2b^2}{2} + k_3b\right) - \left(\frac{k_1a^3}{3} + \frac{k_2a^2}{2} + k_3a\right)$$

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
`1 2 3 -1 1`	`6.666666666667` `6.666666666667`
`1 2 3 -1 0`	`2.333333333333` `2.333333333333`

Hint: นำโค้ดในโปรแกรมข้อ cal_integration มาดัดแปลง

(1) ลูป for(i=0;i<=n-1;i++) เปลี่ยนเป็นลูปที่รันตั้งแต่ i=0,1,…,n

(2) ในลูป for ใช้ if-else ทำการกำหนดค่าตัวแปร coef ซึ่งจะมีค่า 1.0/3.0 ในกรณี i=0 หรือ i=n และมีค่า 2.0/3.0 ในกรณี i=1, 3, 5, 7.. และมีค่า 4.0/3.0 ในกรณี i=2, 4, 6, 8..

(3) การคำนวณค่า sum เปลี่ยนเป็น sum = sum + (y*c*coef) มีการคูณค่า coef ซึ่งจะมีค่าเป็น 1/3 หรือ 2/3 หรือ 4/3 แล้วแต่กรณี

2. Problem name: `cos_approx`

จงเขียนโปรแกรมที่ทำการคำนวณค่าประมาณของ cos(x/2)+cos(x)+cos(2x)+cos(4x) โดยที่ x <u>มีหน่วยเป็น radians</u> โดยกำหนดให้ cos(x) คำนวณใช้สูตร summation ดังสมการข้างล่าง

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots\ldots + (-1)^n \frac{x^{(2n)}}{(2n)!} = \sum_{k=0}^{n} (-1)^k \frac{x^{(2k)}}{(2k)!}$$

เทอมที่ 0 เทอมที่ 1 เทอมที่ n

และกำหนดให้ π radians=180 degree และ π = 3.1416

โปรแกรมจะรับอินพุต 2 จำนวนคั่นด้วยช่องว่างในบรรทัดเดียวกัน โดยอินพุตแรกคือค่าแบบ double สำหรับ x <u>มีหน่วยเป็น degree</u> ส่วนอินพุตตัวที่สองคือค่าแบบ integer สำหรับ n ซึ่งจะคือเทอมลำดับสุดท้ายที่จะใช้ในการคำนวณหาค่าประมาณของ cos(x) ตามสูตรข้างบน

หลังจากนั้นโปรแกรมจะต้องแสดงค่าประมาณ cos(x/2)+cos(x)+cos(2x)+cos(4x) ที่คำนวณได้ออกสู่หน้าจอโดยให้แสดงเลขทศนิยม 6 ตำแหน่ง

Note: x ในสูตรการประมาณค่า cos มีหน่วยเป็น radians แต่โปรแกรมจะรับค่าอินพุตเป็นหน่วย degree ดังนั้นจึงต้องแปลงค่าที่รับมาจากอินพุตให้เป็นอยู่ในรูปของหน่วย radians ก่อนนำไปคำนวณ

นอกจากนี้เพื่อทำให้โปรแกรมดูเป็น module กำหนดให้การเขียนโปรแกรมจะต้องประกอบด้วยการเขียนและเรียกใช้ฟังก์ชัน MyCos(double x, int n) ซึ่งจะรับอาร์กิว์เมนต์ 2 ตัวคือ x ซึ่งมีหน่วยเป็น radian และ n คือค่าลำดับเทอมสุดท้ายที่จะคำนวณค่า cos

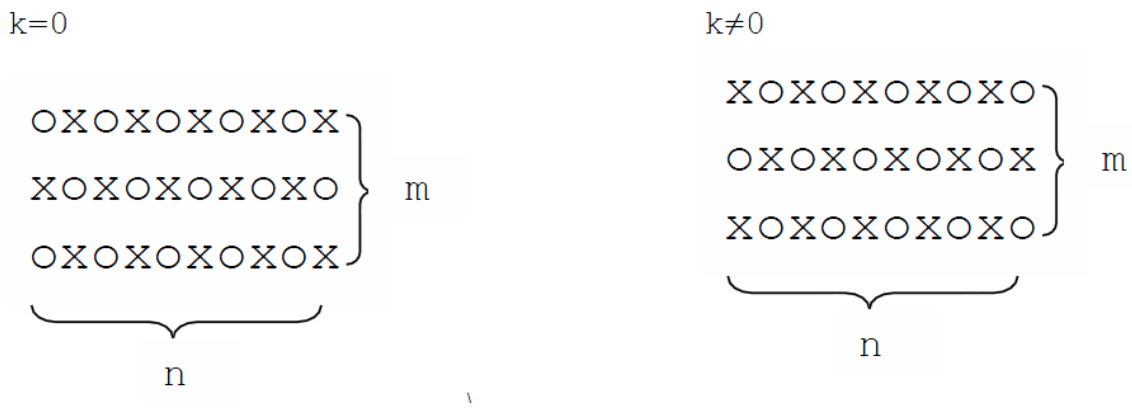
```
double MyCos(double x, int n)
{

}
```

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
90 20	0.707102
90 8	0.740008
45 20	0.630981
30 100	1.831944
180 20	1.000006

3. Problem name: `OX_Figure`

จงเขียนโปรแกรมเพื่อรับค่าจำนวนเต็ม 3 ตัวสมมติว่าเก็บในตัวแปร n m และ k ตามลำดับ หลังจากนั้นโปรแกรมจะต้องพิมพ์เครื่องหมาย O และ X สลับไปมาเป็นรูปสี่เหลี่ยมขนาดกว้าง n สูง m **<u>โดยที่ถ้าค่า k ที่ค่าเท่ากับ 0 มุมบนซ้ายสุดเป็นตัว O แต่ถ้าค่า k ที่ค่าไม่เท่ากับ 0 มุมบนซ้ายสุดเป็นตัว X</u>** ดังตัวอย่างด้านล่างนี้



ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)

`10 3 0`↵	`OXOXOXOXOX`
	`XOXOXOXOXO`
	`OXOXOXOXOX`

`5 5 10`↵	`XOXOX`
	`OXOXO`
	`XOXOX`
	`OXOXO`
	`XOXOX`

4. Problem name: `sum_of_exp_gt_N`

จงเขียนโปรแกรมภาษา C ทำการรับค่าตัวเลขจำนวนบวกชนิด double 1 ตัว จากผู้ใช้ โดยสมมติว่าชื่อ N หลังจากนั้นให้ทำการคำนวณหา***<u>ค่า k ที่น้อยที่สุด</u>***ที่ทำให้

$$e^{(\frac{1}{2})}+e^{(\frac{2}{3})}+e^{(\frac{3}{4})}+e^{(\frac{4}{5})}+e^{(\frac{5}{6})}+.....+e^{(\frac{k}{k+1})}>N$$

หลังจากนั้นให้แสดงค่า k และผลรวม $e^{1/2}+e^{2/3}+e^{3/4}+e^{4/5}+e^{5/6}+.....+e^{k/k+1}$ (ทศนิยม 6 หลัก) ที่โปรแกรมหามาได้ออกสู่หน้าจอ

โดยกำหนดให้การคำนวณค่า e^x ให้ใช้ standard library function exp() ซึ่งมี function declaration ดังนี้

Mathematical Library Functions (require the `math.h` header file)

`double exp(double)`	Returns e raised to its double-precision argument.

ตัวอย่างการใช้งานฟังก์ชัน exp เช่น

```
int main()
{
double x,y;
     x=1.23;
     y=exp(x);  /* y = e^x = e^1.23 */
}
```

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
11	6 12.596391
500	189 501.420911

<u>หมายเหตุ</u>

(i) ใน test case แรก N=11 เราจะพบว่า k ที่น้อยที่สุดคือ 6 และผลรวมคือ 12.596391

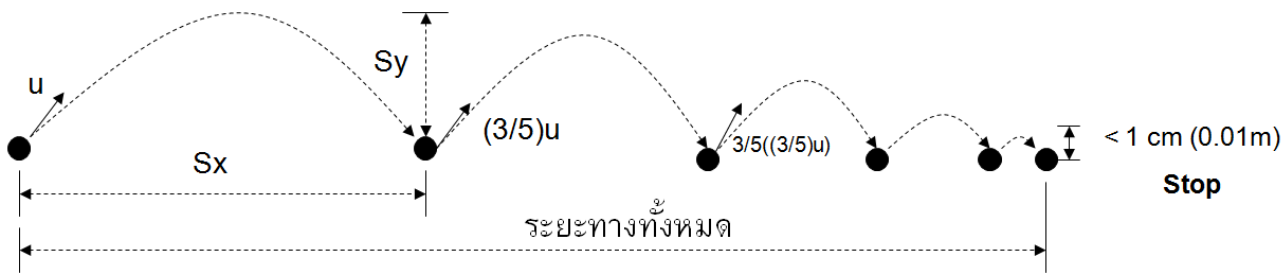
(ii) ใน test case ที่สอง N=500 เราจะพบว่า k ที่น้อยที่สุดคือ 189 และผลรวมคือ 501.420911

5. Problem name: `projectile`

เมื่อขว้างลูกบอลให้เคลื่อนที่แบบวิถีโค้ง Projectile จากพื้นสู่พื้น ด้วยความเร็วเริ่มต้น u ลูกบอลจะเคลื่อนที่แล้วตกกระทบพื้น ด้วยระยะแนวดิ่ง Sy และระยะแนวราบ Sx โดยความสัมพันธ์ทางคณิตศาสตร์[1] (ดูรูปข้างล่างประกอบคำอธิบาย)

$$Sx = u^2/10$$
$$Sy = u^2/40$$

หลังจากลูกบอลตกกระทบพื้นแล้ว **<u>ลูกบอลจะกระดอนขึ้นและเคลื่อนที่เป็นแบบ Projectile ด้วยความเร็วที่เป็น 3/5 เท่าของความเร็วในการเคลื่อนที่แบบ Projectile ในครั้งก่อนหน้านี้</u>** (ดูรูปข้างล่างประกอบคำอธิบาย) แ**<u>ละเมื่อใดก็ตามที่ลูกบอลกระดอนขึ้นด้วยระยะทางในแนวดิ่ง (Sy) ต่ำกว่า 1 เซนติเมตร (0.01 เมตร)</u>** ลูกบอลจะหยุดและไม่กระดอนขึ้นอีก



จงเขียนโปรแกรมเพื่อคำนวณหาว่าเมื่อขว้างลูกบอลให้เคลื่อนที่แบบ Projectile ด้วยความเร็วเริ่มต้น u (หน่วย meters/second) ลูกบอลจะหยุดที่ระยะทางในแนวราบเทียบจากจุดที่ขว้างเป็นระยะเท่าใด และคำนวณหาจำนวนครั้งที่ลูกบอลตกกระทบพื้นก่อนที่ลูกบอลจะหยุด

กำหนดให้โปรแกรมนี้รับค่าความเร็วเริ่มต้น u (หน่วย meters/second) นี้จากผู้ใช้ผ่านคีย์บอร์ด หลังจากนั้นเมื่อโปรแกรมคำนวณหาค่าที่เราต้องการได้แล้ว ให้แสดงค่าระยะทาง (หน่วย meters เลขทศนิยม 4 หลัก) ในบรรทัดถัดไป และแสดงค่าจำนวนครั้งที่ลูกบอลจะตกกระทบพื้นในบรรทัดถัดไป

---

[1] สูตรข้างบนดัดแปลงจาก สูตรในการคำนวณการเคลื่อนที่แบบ Projectile

$$Sx = u^2 \sin 2\Theta / g$$
$$Sy = u^2 (\sin \Theta)^2 / 2g$$

โดยที่เรากำหนด $\Theta = 45^\circ$ และ g=10

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
2	0.6145 4
20	62.4824 8

หมายเหตุ

(i) ใน test case แรก เมื่อขว้างลูกบอลออกไปด้วยความเร็ว 2 m/s ลูกบอลจะหยุดที่ระยะทาง 0.6145 เมตร และตกกระทบพื้นทั้งหมด 4 ครั้ง

(ii) ใน test case ที่สอง เมื่อขว้างลูกบอลออกไปด้วยความเร็ว 20 m/s ลูกบอลจะหยุดที่ระยะทาง 62.4824 เมตร และตกกระทบพื้นทั้งหมด 8 ครั้ง

6. Problem name: `Euclid_GCD`

(เรียบเรียงจาก Schaum's outline series: Programming with C++, 2nd edition, J.R. Hubbard)

ขั้นตอนวิธีการหาตัวหารร่วมมาก หรือ ห.ร.ม. (Greatest Common Divisor or GCD) แบบวิธี Euclid สามารถอธิบายได้ดังนี้

(i) สมมติว่าตัวเลขที่จะหา หรม คือ m และ n โดยที่ m > n

(ii) อัลกอริทึมจะทำการแปลงคู่ของตัวเลข (m,n) ให้กลายเป็น (d,0) โดยการแทนค่า m ด้วย n และพร้อมๆกันแทนค่าตัวเลข n ด้วยค่าของเศษของการหาร m%n และเมื่อใดก็ตามที่เศษของการหารคือ 0 นั่นคือจะได้คู่ตัวเลข (d,0) จะพบว่า d คือ GCD ของ m และ n

<u>ตัวอย่าง 1</u> m=532, n=112 ลำดับการแปลงจาก (532,112) ไปยังตัวเลข (d,0) แสดงได้ดังนี้

- (532,112) -> (112,84) Note: 531%112=84
- (112,84) -> (84,28) Note: 112%84=24
- (84,28) -> (28,0) Note: 84%28=0 จบ เพราะได้คู่ตัวเลขในรูป (d,0) และ d=28 คือ GCD ของ 532 และ 112

<u>ตัวอย่าง 2</u> m=85, n=136 ลำดับการแปลงจาก (85,136) ไปยังตัวเลข (d,0) แสดงได้ดังนี้

- (85,136) -> (136,85) สลับให้ m=136 และ n=85
- (136,85) -> (85,51) Note: 136%85=51
- (85,51) -> (51,34) Note: 85%51=34
- (51,34) -> (34,17) Note: 51%34=17
- (34,17) -> (17,0) Note: 34%17=0 จบ เพราะได้คู่ตัวเลขในรูป (d,0) และ d=17 คือ GCD ของ 85 และ 136 คือ 17

Note: จะเห็นได้ว่าอัลกอริทึมในการหา GCD แบบ Euclid จะทำงานได้เร็วกว่าการค้นหาตัวเลขที่มากสุดที่หารด้วย m และ n ลงตัวซึ่งได้เขียนโปรแกรมมาก่อนหน้านี้

<u>คำสั่ง</u> จงเขียนโปรแกรมเพื่อรับตัวเลขจำนวนเต็มบวกเข้ามา 2 ตัว สมมติคือ m และ n โดยอาจจะเป็นไปได้ในกรณีที่ m>n, m=n หรือ m<n หลังจากนั้นโปรแกรมจะทำการคำนวณและแสดงค่า GCD ของ m และ n โดยใช้วิธี Euclid โดยกำหนดให้เขียนและเรียกใช้ฟังก์ชัน Euclid_GCD ดังนิยามข้างล่าง

```
int Euclid_GCD(int m, int n)
{

}
```

Note: เนื่องจากค่าของ m และ n ที่ส่งมาเป็น argument นั้น m < n หรือ m>n หรือ m=n ได้ดังนั้นส่วนแรกของฟังก์ชันจึงควรทำการสลับค่า m กับ n เพื่อทำให้ m > n แล้วจึงทำการแปลงคู่ตัวเลขโดยใช้วิธีข้างต้นเพื่อหาค่า GCD ของ m และ

ตัวอย่างการทำงานของโปรแกรม

ข้อมูลนำเข้า (Input)	ข้อมูลส่งออก (Output)
49 84	7
125 75	25
532 112	28
112 532	28
85 136	17