คำนวณหาค่า Calories ในอาหารของลูกค้า

ข้อมูล

ณ ร้านอาหารนานาชาติเปิดใหม่แห่งหนึ่ง มีเมนูพิซซ่า 5 อย่างต่อไปนี้

รหัสของรายการอาหาร	รายการอาหาร	ปริมาณกิโลแคลลอรี่ต่อ 1 สไลด์
P01	Digiorno Pepperoni Pizza	265.3
P02	Digiorno Pizza	246.9
P03	Domino’s Cheese Pizza	256.9
P04	Domino’s Sausage Pizza	272.5
P05	Little Caesar Cheese Pizza	309.3

ร้านอาหารแห่งนี้มีความต้องการคำนวณปริมาณแคลลอรี่ในอาหารของลูกค้าแต่ละคนเพื่อเก็บเป็นข้อมูลในการพัฒนาคุณภาพของอาหารต่อไปในอนาคต และต้องการเรียงลำดับของปริมาณแคลลอรี่ของลูกค้าทั้งหมด n คนจากน้อยไปหามาก (ถ้าปริมาณแคลลอรี่เท่ากันให้เรียงชื่อของลูกค้าตามลำดับพจนานุกรม) และ**ถ้าปริมาณแคลลอรี่ของลูกค้าเป็นเลขทศนิยมให้ปัดเศษทศนิยมขึ้น**

ข้อมูลนำเข้า

บรรทัดแรก ระบุจำนวนลูกค้า n คน

บรรทัดที่สอง ระบุชื่อลูกค้า ระบุรหัสรายการอาหารที่ลูกค้าสั่ง และจำนวนสไลด์ที่ลูกค้าสั่งในรายการนั้นๆ โดยใช้เครื่องหมาย Comma (,) เป็นตัวคั่นกลาง

ข้อมูลส่งออก

แสดงชื่อของลูกค้าและปริมาณแคลลอรี่ทั้งหมดของลูกค้าโดยไม่เรียงลำดับ

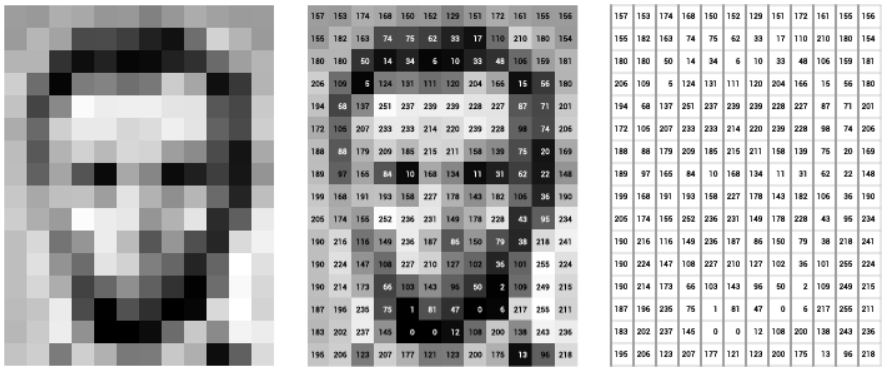
แสดงชื่อของลูกค้าและปริมาณแคลลอรี่ทั้งหมดของลูกค้าโดยเรียงลำดับของปริมาณแคลลอรี่ของลูกค้าทั้งหมด n คนจากน้อยไปหามาก(ถ้าปริมาณแคลลอรี่เท่ากันให้เรียงชื่อของลูกค้าตามลำดับพจนานุกรม)

ตัวอย่าง

Input(จากแป้นพิมพ์)	Output(ทางจอภาพ)
Number of customer: 3	Before ascedning sort

Customer: 1 Kevin,P05,3 Customer: 2 Smith,P05,2 Customer: 3 Jordan,P01,4	[('Kevin', 928), ('Smith', 619), ('Jordan', 1062)] After ascedning sort [('Smith', 619), ('Kevin', 928), ('Jordan', 1062)]
Number of customer: 5 Customer: 1 Aa,P01,2,P03,2 Customer: 2 Ab,P04,3,P03,1 Customer: 3 Ac,P01,2,P03,2 Customer: 4 Ad,P05,4,P03,1 Customer: 5 Ae,P03,1,P05,1	Before ascedning sort [('Aa', 1045), ('Ab', 1075), ('Ac', 1045), ('Ad', 1495), ('Ae', 567)] After ascedning sort [('Ae', 567), ('Aa', 1045), ('Ac', 1045), ('Ab', 1075), ('Ad', 1495)]

ซ่อมภาพ



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

ภาพขาวดำ (Greyscale) ในระบบดิจิตอลจะมีค่าสีในแต่ละพิกเซลอยู่ระหว่าง 0 – 255 โดย 0 แสดงถึงสีดำ และ 255 แสดงถึงสีขาว ค่าที่อยู่ระหว่าง 0 และ 255 คือสีเทาที่มีความเข้มแตกต่างกันไปตามค่าสี

ดอกเตอร์ฟุยฟุยเป็นนักวิทยาศาสตร์ที่ชอบถ่ายภาพขาวดำเป็นงานอดิเรก แต่รูปภาพของเธอเกิดความเสียหายบางส่วน เธอรู้วิธีที่จะซ่อมแซมรูปภาพ แต่ดันเขียนโค๊ดไม่เป็น!!! เธอจึงวานให้น้องช่วยเขียนโค๊ดให้ตามอัลกอริทึมสุดพิลึกของเธอเพื่อซ่อมแซมภาพส่วนที่หายไป

กำหนดให้ภาพมีขนาด n x n พิกเซลเมื่อ n เป็นจำนวนเต็มบวก และ n > 1 ในแต่ละพิกเซลจะมีค่าสีตั้งแต่ 0 – 255 ในกรณีที่พิกเซลนั้นเสียหายจะกำหนดให้มีค่าสีเป็น x อัลกอริทึมสำหรับการซ่อมแซมค่าสีส่วนที่หายไปนั้นเธอจะใช้วิธีสร้างสี่เหลี่ยมขนาด 2x2 พิกเซล วิ่งไปบนภาพโดยเริ่มจากมุมบนซ้ายเพื่อตรวจสอบว่ามีพิกเซลใดที่เสียหายหรือไม่ หากพบว่ามีพิกเซลในสี่เหลี่ยมเสียหาย จะแทนค่าพิกเซลนั้นด้วย**ค่าเฉลี่ยของค่าที่แตกต่างกันทั้งหมดในสี่เหลี่ยมนั้น (ถ้ามีเศษให้ปัดลง)** และมั่นใจว่าจะไม่มีพิกเซลที่เสียหายมากกว่า 1 ตำแหน่งในสี่เหลี่ยมเดียวกัน ตำแหน่งของสี่เหลี่ยมจะขยับไปทางขวาทีละ 1 พิกเซลและเมื่อสุดขอบภาพตำแหน่งของสี่เหลี่ยมจะกลับมาที่ซ้ายสุดของภาพและขยับลงไปจากเดิม 1 พิกเซล อาจมีตำแหน่งของพิกเซลที่เสียหายติดกันมากกว่า 1 ตำแหน่งในภาพ หากใช้วิธีดังกล่าวจะสามารถมั่นใจได้ว่าทุกพิกเซลในภาพสามารถซ่อมแซมได้

ตัวอย่างที่ 1 : ภาพมีขนาด n = 2

ข้อมูลนำเข้า (Input)

127	127
97	**x**

การคำนวน (Calculate)

127 มีซ้ำเลยเอาแค่ตัวเดียว

x = (127 + 97) // 2 = 112

ข้อมูลส่งออก (Output)

127	127
97	**112**

ตัวอย่างที่ 2 : ภาพมีขนาด n = 5

ข้อมูลนำเข้า (Input)	การคำนวน (Calculate)	Step

x	192	84	138	138
97	84	x	x	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

เริ่มต้นที่มุมซ้ายบนสุด

x = (192 + 97 + 84) // 3 = 124

124	192	84	138	138
97	84	x	x	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

124	192	84	138	138
97	84	x	x	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

ขยับมาทางขวา 1 พิกเซล

84 มีซ้ำเลยเอาแค่ตัวเดียว

x = (192 + 84) // 2 = 138

124	192	84	138	138
97	84	**138**	x	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

124	192	84	138	138
97	84	138	x	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

ขยับมาทางขวา 1 พิกเซล

138 มีซ้ำเลยเอาแค่ตัวเดียว

x = (138 + 84) // 2 = 111

124	192	84	138	138
97	84	138	**111**	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

124	192	84	138	138
97	84	138	111	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

ขยับมาทางขวา 1 พิกเซล

สุดขอบพอดีและไม่มีพิกเซล

ที่เสียหาย กลับไปซ้ายสุด

แล้วขยับลงมา 1 พิกเซล

124	192	84	138	138
97	84	138	**111**	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

124	192	84	138	138
97	84	138	111	138
10	x	x	252	138
255	10	10	17	252
121	10	185	192	255

x = (97 + 84 + 10) // 3 = 63

124	192	84	138	138
97	84	138	111	138
10	63	x	252	138
255	10	10	17	252
121	10	185	192	255

124	192	84	138	138
97	84	138	111	138
10	63	x	252	138
255	10	10	17	252
121	10	185	192	255

x = (63 + 84 + 138) // 3 = 95

124	192	84	138	138
97	84	138	111	138
10	63	95	252	138
255	10	10	17	252
121	10	185	192	255

ข้อมูลนำเข้า

บรรทัดแรก ระบุจำนวนเต็ม n แทนจำนวนขนาดของภาพ n x n

บรรทัดที่สอง ระบุสตริง s ที่สมาชิก a_{rc} ที่มีค่าอยู่ระหว่าง 0 – 255 หรือ x ในกรณีที่พิกเซลนั้นเสียหาย จำนวน n x n ตัว $0 \leq r, c \leq n-1$ เมื่อ r แทนตำแหน่งของแถว และ c แทนตำแหน่งของคอลัมน์ โดย r และ c เป็นจำนวนเต็มบวก แต่ละจำนวน a_{rc} ใน s จะคั่นด้วยเว้นวรรค ตัวเลขอาจมีค่าซ้ำกันได้ (เช่น 124 x 84 138 138)

ข้อมูลส่งออก

แสดงผลลัพธ์ของค่าสีในแต่ละพิกเซลที่ผ่านการซ่อมแซมแล้วโดย จำนวนแถวของผลลัพธ์จะมีค่าเท่ากับ n และในแต่ละบรรทัดจะมีข้อมูลค่าสีของแต่ละพิกเซล n จำนวน

ตัวอย่าง

input	output
2 127 127 97 x	127 127 97 112

3 255 0 x 0 x 127 255 x x	255 0 63 0 127 127 255 127 127
4 6 x 47 15 110 25 x x x 36 30 x x 36 32 x	6 47 47 15 110 25 36 32 57 36 30 32 46 36 32 31
5 x 192 84 138 138 97 84 x x 138 10 x x 252 138 255 10 10 17 252 121 10 185 192 255	124 192 84 138 138 97 84 138 111 138 10 63 95 252 138 255 10 10 17 252 121 10 185 192 255

Derivative

การคำนวน Gradient descent ในเรื่อง Multivariant Linear Regression ที่นิสิตได้เคยเรียนมานั้น จำเป็นต้องคำนวนค่า derivative จากค่า loss (MSE) เพื่อหาค่า gradient ในการนำไปปรับ weight ในข้อนี้จะให้นิสิตเขียนโปรแกรมคำนวน derivative โดยใช้ numpy โดยโปรแกรมจะรับข้อมูล training, weight เริ่มต้น, ค่า y (ผลเฉลยของข้อมูล) จากนั้นคืนค่า derivative ออกมา

$$\text{derivative} = 2*(X^t(w(X^t) - y)) / \text{number of samples}$$

ข้อมูลนำเข้า

บรรทัดแรก รับจำนวนเต็ม 2 จำนวน โดยที่ n คือ จำนวน samples และ m คือ จำนวน features
n บรรทัดถัดไป รับจำนวนเต็ม m จำนวน (f1, f2, … ,fm)
บรรทัดถัดไปรับ ค่า y จำนวนเต็ม n จำนวน (y1, y2, … , yn)
บรรทัดสุดท้าย เป็นการรับจำนวนเต็ม m+1 จำนวนเป็น initial weight

ข้อมูลส่งออก

แสดงค่า derivative ที่คำนวณออกมา

ตัวอย่าง

input	output
3 5 1 1 2 1 2 1 2 4 2 4 1 6 3 6 3 3 5 4 1 2 3 4 5	78.000 294.667 247.333 294.667 247.333
3 3 1 3 8 1 5 4 1 6 7 4 4 2 1 8 4	120.667 582.667 774.000