CSE421 — Computer Graphics Midterm Solution (Spring 2026)

Question 1(a) — Anti-Aliasing Techniques [5 marks]

প্রশ্নটা কী বলছে?

একটা CAD application-এ diagonal edges zoom করলে jagged artifacts ও moiré pattern দেখা যায়। তিনটা সমাধান দেওয়া আছে:

1. Rendering resolution বাড়ানো
2. Pixel intensity = edge ও pixel-এর exact overlap area হিসাব করে
3. Dense virtual grid-এ render করে low-pass filter দিয়ে final display করা

এই তিনটা কোন anti-aliasing technique সেটা identify করো এবং যেকোনো দুটোর working principle বোঝাও।

Identification:

Scenario-র কথা	**Technique**
Uniformly increasing rendering resolution	**Increasing Resolution**
Computing pixel intensities based on exact overlap area	**Area Sampling (Weighted/Unweighted)**
Dense virtual grid + low-pass filtering before display	**Supersampling (Post-filtering)**

Technique 1: Increasing Resolution

Definition: Increasing resolution means increasing the number of pixels in both X and Y directions so that each pixel becomes smaller, making jagged edges (jaggies) less visible.

কেন কাজ করে **(Bangla explanation):** Pixel যত ছোট হবে, রেখার প্রান্তে ভাঙা-ভাঙা অংশ তত কম চোখে পড়বে। কারণ ছোট pixel মানে বেশি detail ধরা যায়।

Working Principle:

- Resolution double করলে (X ও Y দুই দিকে) → total pixels হয় **4 times**
- প্রতিটা pixel-এর আকার ছোট হয় → jaggies কম দৃশ্যমান হয়

Limitation:

- Memory cost = **4×** বেশি
- Processing cost = **4×** বেশি
- Aliasing পুরোপুরি দূর হয় না, শুধু কমে
- Practically standalone solution হিসেবে ব্যবহার করা হয় না

Technique 2: Area Sampling (Pre-filtering)

Definition: Area sampling computes the intensity of each pixel based on how much area of that pixel is covered by the object (edge/line) being drawn.

কেন কাজ করে **(Bangla explanation):** একটা pixel হয় পুরো জ্বলবে নয়তো পুরো নিভবে — এই binary নিয়মের বদলে, রেখা pixel-এর কতটুকু ঢেকেছে সেই অনুপাতে pixel-কে আংশিকভাবে জ্বালানো হয়। ফলে প্রান্তে smooth gradient তৈরি হয়।

Working Principle:

Step 1: প্রতিটা pixel-কে একটা ছোট বর্গ হিসেবে কল্পনা করো।

Step 2: রেখাকে ১ pixel প্রস্থের একটা area হিসেবে কল্পনা করো।

Step 3: রেখা ও pixel-এর overlap area হিসাব করো।

Step 4: সেই অনুপাতে intensity নির্ধারণ করো।

উদাহরণ:

- রেখা pixel-এর 75% ঢাকলে → pixel intensity = 75%
- রেখা pixel-এর 25% ঢাকলে → pixel intensity = 25%

Two types:

- **Unweighted:** শুধু coverage দেখে
- **Weighted:** Coverage + pixel center থেকে দূরত্ব দুটোই দেখে → আরও smooth result

---

Technique 3: Supersampling (Post-filtering)

Definition: Supersampling renders the image at a higher resolution (virtual grid), applies a low-pass filter to remove high frequencies, then downsamples to the actual display resolution.

কেন কাজ করে **(Bangla explanation):** প্রথমে ছবিটা অনেক বড় resolution-এ তৈরি করা হয়। তারপর low-pass filter দিয়ে jaggies তৈরিকারী high frequency গুলো কেটে ফেলা হয়। তারপর সেটা আসল resolution-এ নামিয়ে আনা হয়। ছোট করার সময় আশেপাশের pixel-এর রঙ average হয়ে smooth দেখায়।

Working Principle (3 Steps):

Step 1 — Supersample: Continuous image-কে n গুণ বড় resolution-এ sample করো → এটা Virtual Image।

Step 2 — Filter: Virtual image-এ Low-pass filter প্রয়োগ করো → high frequency (jagged edges) কেটে ফেলো।

Step 3 — Downsample: Filtered image-কে actual frame resolution-এ নামিয়ে আনো।

Advantages:

- Hardware-independent
- Symmetric filter use করা যায়
- যেকোনো shape-এ কাজ করে

Limitation:

- Virtual image resolution-এর সীমা আছে
- Aliasing পুরোপুরি দূর হয় না — শুধু Nyquist limit উপরে তোলে

Question 1(b) — 2D Rotation Transformation Matrix [3 marks]

প্রশ্নটা কী বলছে?

Origin-কে center ধরে একটা point-কে θ কোণে rotate করার 2D transformation matrix derive করো।

Derivation:

ধরো একটা point **P(x, y)** আছে, origin থেকে দূরত্ব **r**, এবং X-axis এর সাথে কোণ **φ**।

তাহলে:

$$x = r \cos\phi$$ $$y = r \sin\phi$$

এখন এই point-কে **θ** কোণে rotate করলে নতুন point হবে **P'(x', y')**, যার কোণ হবে **(φ + θ)**:

$$x' = r \cos(\phi + \theta)$$ $$y' = r \sin(\phi + \theta)$$

Trigonometric expansion প্রয়োগ করো**:**

$$x' = r(\cos\phi \cos\theta - \sin\phi \sin\theta)$$ $$y' = r(\sin\phi \cos\theta + \cos\phi \sin\theta)$$

x = r cosφ এবং **y = r sinφ** বসাও**:**

$$x' = x\cos\theta - y\sin\theta$$ $$y' = x\sin\theta + y\cos\theta$$

Matrix আকারে লেখো**:**

$$\begin{bmatrix} x' \ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \ y \end{bmatrix}$$

সুতরাং **2D Rotation Matrix:**

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \ \sin\theta & \cos\theta \end{bmatrix}$$

সহজ কথায় **(Bangla):** কোনো point-কে origin-এর চারদিকে θ কোণে ঘোরাতে হলে, তার X ও Y coordinate-এ উপরের matrix দিয়ে গুণ করলেই নতুন rotated position পাওয়া যায়।

Question 2 — Bresenham's Line Drawing Algorithm [7 marks]

প্রশ্নটা কী বলছে**?**

V-shaped figure আঁকতে হবে তিনটা point দিয়ে:

- **A(38, 45), B(40, 40), C(42, 45)**
- Line segment AB এবং BC — দুটো আলাদাভাবে Bresenham's algorithm apply করো।

Line AB: A(38,45) → B(40,40)

Step 1 — Values বের করো**:**

- Δx = 40 - 38 = **2**
- Δy = 40 - 45 = **-5** → |Δy| = 5
- যেহেতু |Δy| > |Δx|, এটা **steep line** → Y-axis along drive করতে হবে

Steep line-এ **swap** করো**:** x ও y swap করে হিসাব করবো, তারপর আবার swap করে plot করবো।

Swapped: A(45, 38) → B(40, 40)

- $\Delta x' = 40 - 45 = -5$ → Step direction: Y decreasing
- $\Delta y' = 40 - 38 = 2$

যেহেতু Y কমছে: প্রতি step-এ y থেকে 1 বাদ দেবো।

- **$2\Delta x' = 4$**
- **$2\Delta y' - 2\Delta x' = 2(2) - 2(5) = -6$**
- **Initial decision parameter $p_0 = 2\Delta y' - \Delta x' = 2(2) - 5 = -1$**

Step	y (driving)	x	p	Plot (x,y)
0	45	38	-1	**(38, 45)**
1	44	38	-1+4= **3**	**(38, 44)**
2	43	39	3-6= **-3**	**(39, 43)**
3	42	39	-3+4= **1**	**(39, 42)**
4	41	39	1-6= **-5**	**(39, 41)**
5	40	40	—	**(40, 40)**

Rule:

- $p < 0$ হলে → x same রাখো, $p_next = p + 2\Delta y'$
- $p \geq 0$ হলে → x এক step বাড়াও (বা কমাও direction অনুযায়ী), $p_next = p + 2\Delta y' - 2\Delta x'$

AB-এর pixel points: (38,45), (38,44), (39,43), (39,42), (39,41), (40,40)

Line BC: B(40,40) → C(42,45)

Step 1 — Values বের করো**:**

- $\Delta x = 42 - 40 =$ **2**
- $\Delta y = 45 - 40 =$ **5**
- যেহেতু $|\Delta y| > |\Delta x|$, এটাও **steep line** → Y-axis drive করবে

Swap করে: B(40, 40) → C(45, 42)

- $\Delta x' = 5$ (Y বাড়ছে, প্রতি step y+1)
- $\Delta y' = 2$
- $\mathbf{2\Delta y' = 4}$
- $\mathbf{2\Delta y' - 2\Delta x' = 4 - 10 = -6}$
- $\mathbf{p_0 = 2\Delta y' - \Delta x' = 4 - 5 = -1}$

Step	y (driving)	x	p	Plot (x,y)
0	40	40	-1	**(40, 40)**
1	41	40	-1+4= **3**	**(40, 41)**
2	42	41	3-6= **-3**	**(41, 42)**
3	43	41	-3+4= **1**	**(41, 43)**
4	44	41	1-6= **-5**	**(41, 44)**
5	45	42	—	**(42, 45)**

BC-এর pixel points: (40,40), (40,41), (41,42), (41,43), (41,44), (42,45)

Question 3 — Lens-Shaped Figure using Midpoint Circle Drawing Algorithm

Given Information:

- Two circles, each with **radius r = 10**
- Circle 1 center: **(6, 0)**
- Circle 2 center: **(-6, 0)**
- Inner overlapping portions trimmed → only **outer arcs** remain

---

Step 1: Apply Midpoint Circle Algorithm (origin ধরে, r = 10)

Initial values:

- x = 0, y = 10
- p_0 = 1 - r = 1 - 10 = **-9**

Rules:

- If p < 0 → x+1, y same, p_next = p + 2x_new + 1
- If p ≥ 0 → x+1, y-1, p_next = p + 2x_new - 2y_new + 1

Table:

Step	x	y	p
0	0	10	-9
1	1	10	-9+2(1)+1 = **-6**
2	2	10	-6+2(2)+1 = **-1**
3	3	10	-1+2(3)+1 = **6**

4	4	9	6+2(4)-2(9)+1 = **-1**
5	5	9	-1+2(5)+1 = **10**
6	6	8	10+2(6)-2(8)+1 = **7**
7	7	7	7+2(7)-2(7)+1 = **8**
8	8	6	stop (x > y)

1st Octant Points (origin-based): (0,10), (1,10), (2,10), (3,10), (4,9), (5,9), (6,8), (7,7)

Step 2: Full Circle Points via 8-way Symmetry

প্রতিটা (x, y) থেকে 8টা symmetric point: **(±x, ±y) and (±y, ±x)**

(x,y)	**All 8 Symmetric Points**
(0,10)	(0,10),(0,-10),(10,0),(-10,0)
(1,10)	(1,10),(1,-10),(-1,10),(-1,-10),(10,1),(10,-1),(-10,1),(-10,-1)
(2,10)	(2,10),(2,-10),(-2,10),(-2,-10),(10,2),(10,-2),(-10,2),(-10,-2)
(3,10)	(3,10),(3,-10),(-3,10),(-3,-10),(10,3),(10,-3),(-10,3),(-10,-3)
(4,9)	(4,9),(4,-9),(-4,9),(-4,-9),(9,4),(9,-4),(-9,4),(-9,-4)

(5,9)	(5,9),(5,-9),(-5,9),(-5,-9),(9,5),(9,-5),(-9,5),(-9,-5)
(6,8)	(6,8),(6,-8),(-6,8),(-6,-8),(8,6),(8,-6),(-8,6),(-8,-6)
(7,7)	(7,7),(7,-7),(-7,7),(-7,-7)

Step 3: Shift to Actual Centers

Circle 1 → Center (6, 0): add (+6, 0) to all points

Origin Point	Circle 1 Point	Origin Point	Circle 1 Point
(0,10)	**(6,10)**	(0,-10)	**(6,-10)**
(10,0)	**(16,0)**	(-10,0)	**(-4,0)**
(1,10)	**(7,10)**	(1,-10)	**(7,-10)**
(-1,10)	**(5,10)**	(-1,-10)	**(5,-10)**
(10,1)	**(16,1)**	(10,-1)	**(16,-1)**
(-10,1)	**(-4,1)**	(-10,-1)	**(-4,-1)**
(2,10)	**(8,10)**	(2,-10)	**(8,-10)**
(-2,10)	**(4,10)**	(-2,-10)	**(4,-10)**

(10,2)	**(16,2)**	(10,-2)	**(16,-2)**
(-10,2)	**(-4,2)**	(-10,-2)	**(-4,-2)**
(3,10)	**(9,10)**	(3,-10)	**(9,-10)**
(-3,10)	**(3,10)**	(-3,-10)	**(3,-10)**
(10,3)	**(16,3)**	(10,-3)	**(16,-3)**
(-10,3)	**(-4,3)**	(-10,-3)	**(-4,-3)**
(4,9)	**(10,9)**	(4,-9)	**(10,-9)**
(-4,9)	**(2,9)**	(-4,-9)	**(2,-9)**
(9,4)	**(15,4)**	(9,-4)	**(15,-4)**
(-9,4)	**(-3,4)**	(-9,-4)	**(-3,-4)**
(5,9)	**(11,9)**	(5,-9)	**(11,-9)**
(-5,9)	**(1,9)**	(-5,-9)	**(1,-9)**
(9,5)	**(15,5)**	(9,-5)	**(15,-5)**
(-9,5)	**(-3,5)**	(-9,-5)	**(-3,-5)**

(6,8)	**(12,8)**	(6,-8)	**(12,-8)**
(-6,8)	**(0,8)**	(-6,-8)	**(0,-8)**
(8,6)	**(14,6)**	(8,-6)	**(14,-6)**
(-8,6)	**(-2,6)**	(-8,-6)	**(-2,-6)**
(7,7)	**(13,7)**	(7,-7)	**(13,-7)**
(-7,7)	**(-1,7)**	(-7,-7)	**(-1,-7)**

Circle 2 → Center (-6, 0): add (-6, 0) to all points

Origin Point	Circle 2 Point	Origin Point	Circle 2 Point
(0,10)	**(-6,10)**	(0,-10)	**(-6,-10)**
(10,0)	**(4,0)**	(-10,0)	**(-16,0)**
(1,10)	**(-5,10)**	(1,-10)	**(-5,-10)**
(-1,10)	**(-7,10)**	(-1,-10)	**(-7,-10)**
(10,1)	**(4,1)**	(10,-1)	**(4,-1)**
(-10,1)	**(-16,1)**	(-10,-1)	**(-16,-1)**

(2,10)	**(-4,10)**	(2,-10)	**(-4,-10)**
(-2,10)	**(-8,10)**	(-2,-10)	**(-8,-10)**
(10,2)	**(4,2)**	(10,-2)	**(4,-2)**
(-10,2)	**(-16,2)**	(-10,-2)	**(-16,-2)**
(3,10)	**(-3,10)**	(3,-10)	**(-3,-10)**
(-3,10)	**(-9,10)**	(-3,-10)	**(-9,-10)**
(10,3)	**(4,3)**	(10,-3)	**(4,-3)**
(-10,3)	**(-16,3)**	(-10,-3)	**(-16,-3)**
(4,9)	**(-2,9)**	(4,-9)	**(-2,-9)**
(-4,9)	**(-10,9)**	(-4,-9)	**(-10,-9)**
(9,4)	**(3,4)**	(9,-4)	**(3,-4)**
(-9,4)	**(-15,4)**	(-9,-4)	**(-15,-4)**
(5,9)	**(-1,9)**	(5,-9)	**(-1,-9)**
(-5,9)	**(-11,9)**	(-5,-9)	**(-11,-9)**

(9,5)	**(3,5)**	(9,-5)	**(3,-5)**
(-9,5)	**(-15,5)**	(-9,-5)	**(-15,-5)**
(6,8)	**(0,8)**	(6,-8)	**(0,-8)**
(-6,8)	**(-12,8)**	(-6,-8)	**(-12,-8)**
(8,6)	**(2,6)**	(8,-6)	**(2,-6)**
(-8,6)	**(-14,6)**	(-8,-6)	**(-14,-6)**
(7,7)	**(1,7)**	(7,-7)	**(1,-7)**
(-7,7)	**(-13,7)**	(-7,-7)	**(-13,-7)**

Step 4: Trimming — Inner Overlapping Parts বাদ দেওয়া

Boundary বের করো**:** $$x_{boundary} = \frac{6 + (-6)}{2} = \frac{0}{2} = 0$$

তাহলে boundary line = **x = 0**

*(*বাংলায় বোঝার জন্য*:* দুই *center 6* আর *-6,* এদের ঠিক মাঝখান হলো *x=0* মানে *Y-axis*। *Circle 1* ডানে আছে*, Circle 2* বামে আছে। তাই *Circle 1*-এর যেসব *point* বামে *(x≤0)* চলে গেছে সেগুলো *Circle 2*-এর ভেতরে ঢুকে গেছে — বাদ। একইভাবে *Circle 2*-এর যেসব *point* ডানে *(x≥0)* চলে গেছে সেগুলো *Circle 1*-এর ভেতরে — বাদ।*)*

Rule:

- Circle 1-এর points → **x ≤ 0** হলে বাদ (keep only x > 0)
- Circle 2-এর points → **x ≥ 0** হলে বাদ (keep only x < 0)

Circle 1 — Final Kept Points (x > 0 only):

(6,10), (6,-10), (16,0), (7,10), (7,-10), (5,10), (5,-10), (16,1), (16,-1), (8,10), (8,-10), (4,10), (4,-10), (16,2), (16,-2), (9,10), (9,-10), (3,10), (3,-10), (16,3), (16,-3), (10,9), (10,-9), (2,9), (2,-9), (15,4), (15,-4), (11,9), (11,-9), (1,9), (1,-9), (15,5), (15,-5), (12,8), (12,-8), (14,6), (14,-6), (13,7), (13,-7)

Circle 2 — Final Kept Points (x < 0 only):

(-6,10), (-6,-10), (-16,0), (-5,10), (-5,-10), (-7,10), (-7,-10), (-16,1), (-16,-1), (-4,10), (-4,-10), (-8,10), (-8,-10), (-16,2), (-16,-2), (-3,10), (-3,-10), (-9,10), (-9,-10), (-16,3), (-16,-3), (-2,9), (-2,-9), (-10,9), (-10,-9), (-15,4), (-15,-4), (-1,9), (-1,-9), (-11,9), (-11,-9), (-15,5), (-15,-5), (-12,8), (-12,-8), (-14,6), (-14,-6), (-13,7), (-13,-7)

Final Answer Summary

- **Algorithm used:** Midpoint Circle Drawing Algorithm, r = 10
- **1st Octant points (origin):** (0,10), (1,10), (2,10), (3,10), (4,9), (5,9), (6,8), (7,7)
- **Circle 1** center (6,0) → kept only **x > 0** points (right outer arc)
- **Circle 2** center (-6,0) → kept only **x < 0** points (left outer arc)
- Together these two outer arcs form the **lens shape**

হ্যাঁ, "plot the tentative shape" মানে হলো points গুলো grid-এ বসিয়ে একটা rough shape দেখানো। Exam-এ এটা করার সহজ উপায়:

Exam-এ কীভাবে আঁকবে?

ধাপ ১**:** একটা grid আঁকো (X ও Y axis সহ)

ধাপ ২**:** Circle 1 (center 6,0) এর kept points গুলো dot দিয়ে বসাও — শুধু ডান দিকের **arc**

ধাপ ③**:** Circle 2 (center -6,0) এর kept points গুলো dot দিয়ে বসাও — শুধু বাম দিকের **arc**

ধাপ ৪**:** Dots গুলো curve দিয়ে connect করো

Tentative মানে কী?

Tentative = আনুমানিক/খসড়া। মানে exact perfect circle না হলেও চলবে। Algorithm থেকে পাওয়া pixel points গুলো বসিয়ে যে rough shape দেখা যায় সেটাই tentative shape।

Exam-এ practically কী লিখবে?

পুরো ৫০+ points plot করা সম্ভব না সময়ের মধ্যে। তাই exam-এ যা করবে:

১. Algorithm-এর table দেখাও ✅ (already done)

২. Grid-এ শুধু **key points** বসাও — যেমন:

- Circle 1 এর: (16,0), (6,10), (6,-10), (12,8), (12,-8) — এগুলো দিয়েই arc বোঝা যায়
- Circle 2 এর: (-16,0), (-6,10), (-6,-10), (-12,8), (-12,-8)

৩. Points গুলো মিলিয়ে **smooth curve** টানো

৪. Shape টা দেখতে হবে এরকম — বাম দিকে একটা C-shape arc, ডান দিকে একটা উল্টো C-shape arc, মিলে lens/crescent shape।

> 📌 সহজ কথায়: চবি আঁকা মানে সব point একদম perfect বসানো না। Grid-এ কয়েকটা key point বসাও, তারপর হাতে smooth curve টেনে দাও — এটাই examiner দেখতে চায়। Shape টা lens-এর মতো দেখালেই marks পাবে।

Question 4 — 3D X-axis Rotation, Find θ [3 marks]

প্রশ্নটা কী বলছে?

Initial coordinates: **A(-2,3,-1), B(1,3,-1), C(1,-2,3)** After rotating θ degrees about X-axis: New coordinates: **A'(-2,3,-1), B'(1,3,-1), C'(1,-2,3)**

লক্ষ্য করো — আগের ও পরের **coordinates** একই! θ-এর মান বের করো।

X-axis Rotation Matrix:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \ 0 & \cos\theta & -\sin\theta \ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

X-axis rotation-এ x-coordinate change হয় না। শুধু y ও z বদলায়:

$$y' = y\cos\theta - z\sin\theta$$ $$z' = y\sin\theta + z\cos\theta$$

Point A(-2, 3, -1) থেকে **A'(-2, 3, -1)** এর জন্য:

$$y' = 3\cos\theta - (-1)\sin\theta = 3\cos\theta + \sin\theta = 3 \quad ...(i)$$ $$z' = 3\sin\theta + (-1)\cos\theta = 3\sin\theta - \cos\theta = -1 \quad ...(ii)$$

Equation (i) থেকে: $$3\cos\theta + \sin\theta = 3$$

Equation (ii) থেকে: $$3\sin\theta - \cos\theta = -1$$

θ = 0° check করি:

- 3cos(0°) + sin(0°) = 3(1) + 0 = **3** ✓
- 3sin(0°) - cos(0°) = 0 - 1 = **-1** ✓

∴ θ = 0°

সহজ ব্যাখ্যা **(Bangla):** যেহেতু rotate করার আগে ও পরে coordinates একই থেকেছে, মানে কোনো পরিবর্তন হয়নি। X-axis rotation matrix-এ θ = 0° বসালে matrix হয়ে যায় identity matrix — যা কোনো পরিবর্তন করে না। তাই **θ = 0°**।

> 📌 **Exam-**এ লেখার **tip:** প্রতিটা question-এ আগে formula/definition দাও, তারপর step-by-step calculation করো, শেষে answer clearly বোলো। Table আকারে calculation দেখালে marks বেশি পাওয়া যায়।

তুমি একদম সঠিক কথা বলেছো! 7 marks-এর জন্য এত কিছু করা practically impossible exam-এ।

Exam-এ 7 marks-এর জন্য আসলে কী করতে হবে?

Marking breakdown টা এরকম হয় সাধারণত:

অংশ	কী করবে	Marks
Algorithm table	1st octant-এর points বের করো	~3
Center-এ shift	Points-এ center যোগ করো (কিছু examples দেখাও, সব না)	~2
Trimming explain	Boundary বের করো, rule বলো	~1
Plot	Rough shape আঁকো	~1

তাহলে Exam-এ shortcut কী?

১. **Algorithm table** — এটা পুরো করতেই হবে, বাদ দেওয়ার উপায় নেই।

২. **Center shift** — সব points লেখার দরকার নেই। শুধু **2-3** টা **example** দেখাও এভাবে:

> "Adding center (6,0) to all points, e.g. (0,10)→(6,10), (7,7)→(13,7) and so on."

৩. **Trimming** — শুধু rule টা লেখো:

"Boundary = x=0. Circle 1: keep x>0 only. Circle 2: keep x<0 only."

৪. **Plot** — 4-5 টা key point বসাও, curve টানো। 2 মিনিটের কাজ।

📌 মূল কথা: Examiner দেখতে চায় তুমি algorithm জানো কিনা। সব points লেখা না থাকলেও marks কাটে না, যদি method টা সঠিক থাকে। Table + rule + rough plot — এই তিনটা থাকলেই 7-এ 6+ পাবে।