Student #:
Name:

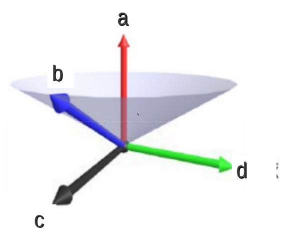
Write down answers in-between questions. Please answer using short sentences. The given spaces should be more than enough.

1. How many bytes are necessary to store a 1024×1024 color image with an alpha channel using 8 bits per channel ?

2. What is a parametric form (or explicit equation) for the axis-aligned 2D ellipse of which center is at $\boldsymbol{p}$, width and height are a, b ? (hint: use parameter $t \in [0, 2\pi)$. e.g., { f(t) | $t \in [0, 2\pi)$ })

3. Write down the 4×4 rotation matrix M that maps the orthonormal 3D vectors $\boldsymbol{u} = (x_u, y_u, z_u), \boldsymbol{v} = (x_v, y_v, z_v)$, and $\boldsymbol{w} = (x_w, y_w, z_w)$, to orthonormal 3D vectors $\boldsymbol{a} = (x_a, y_a, z_a), \boldsymbol{b} = (x_b, y_b, z_b)$, and $\boldsymbol{c} = (x_c, y_c, z_c)$, so $M\boldsymbol{u} = \boldsymbol{a}, M\boldsymbol{v} = \boldsymbol{b}$, and $M\boldsymbol{w} = \boldsymbol{c}$.

4. Represent vector $\boldsymbol{c}$ and $\boldsymbol{d}$ in terms of vector $\boldsymbol{a}$ and $\boldsymbol{b}$ using the cross product operator (×) and the length operator (| |). Vector $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ are in the same plane, $\boldsymbol{d}$ is orthogonal to the other vectors, $\boldsymbol{a}$ is orthogonal to $\boldsymbol{c}$ and $\boldsymbol{d}$, and all vectors are unit vectors.



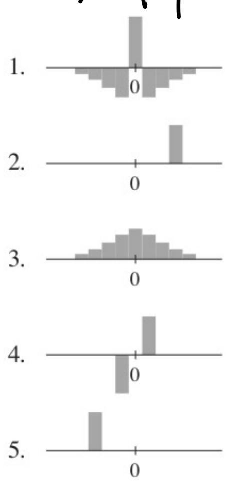
→ typo of j'

5. If the following 1D discrete filters are used to define 2D filters and applied to images, which filter goes with which operation? hints: assuming $a[i,j]=a_1[i]a_1[j]$, a filtering operation can be defined using the convolution operator * as follows:

$$(a*b)[i,j]=\sum_{i',j'} a[i',j']b[i-i',j-j']=\sum_{i'} a_1[i']\left(\sum_{j'} a_1[i']b[i-i',j-j']\right)$$

, where 1D filter a_1 is one of the followings.

simply try 1D convolution of a_1 and signal [0,0,0,0,0,1,1,1,1,1]



$a_1=[0,0,0,0,1]$ ($j=0$ at the middle entry)

(a) blur
(b) sharpen
(c) shift left and down
(d) shift right and up
(e) differentiate (in some way)

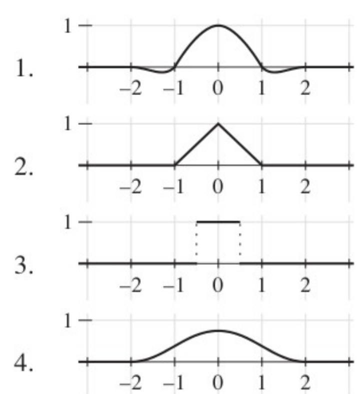
$a_1=[1,0,0,0,0]$ ($j=0$ at the middle entry)

1 – (b) // 중앙값 강조, 주변 평균값 감소
2 – (d) ☆
3 – (a) // 주변 픽셀들의 가중평균
4 – (e)
5 – (c)

6. If we use each of the following 1D reconstruction filters to reconstruct a continuous function $g(x)$ from a sequence of samples $f[i]$ using continuous-discontinuous convolution, for which filters will $g(x)$ be C^0 ?
For which filters will $g(x)$ be C^1 ?
For which filters will $g(x)$ interpolate $f[i]$?
hints: the reconstructed function $g(x)$ is defined as $g(x)=\sum_i f[i]a(x-i)$ for a arbitrary sequence of samples $f[i]$ when each of the followings is $a(x)$.

the C^k continuity of $g(x)$ is the same as that of the filter $a(x)$

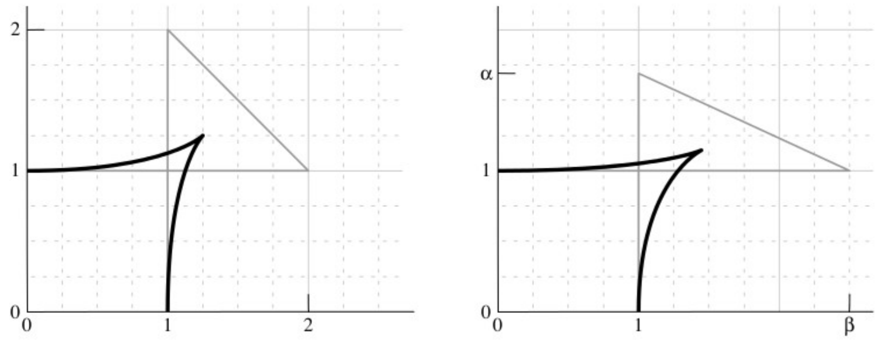
C^0 : 1, 2, 4
C^1 : 1, 4



interpolate : 1, 2, (3) optional

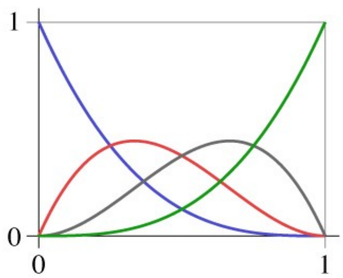
filter needs have value of 1 at 0, and 0 at other integers.

7. Bezier spline with the control points (1, 0), (1, 2), (2, 1), (0, 1) will form a cusp, as shown here on the left:



a. Sketch plots of the coordinate functions x(t) and y(t) and their derivatives. Explain how you can tell that the cusp happens by looking at these plots.
b. What are the parametric (C^k) continuity and geometric (G^k) continuity at the cusp?

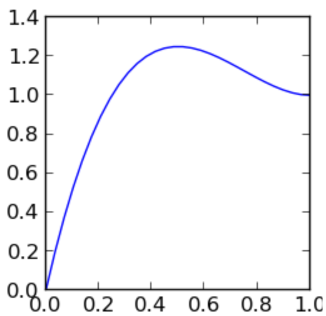
hints: a Bezier spline is defined using matrix [-1 3 -3 1; 3 -6 3 0; -3 3 0 0 ; 1 0 0 0]. Here ";" represents a new line (row separator). The bezier basis functions look like this:



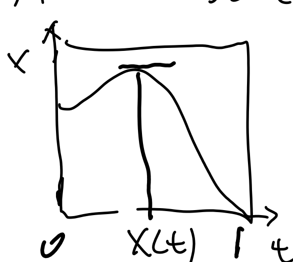
$$(t^3\ \ t^2\ \ t\ \ 1)\begin{pmatrix}-1&3&-3&1\\3&-6&3&0\\-3&3&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}1&0\\1&2\\2&1\\0&1\end{pmatrix}$$

$$=(t^3\ \ t^2\ \ t\ \ 1)\begin{pmatrix}-4&4\\3&-9\\0&6\\1&0\end{pmatrix}=(x(t)\ \ y(t))$$

$4t^3-9t^2+6t$ looks like below. (explain if this plot is relevant or not, and why.)



∴ the above plot is y(t),



and x(t) is a vertically mirrored version of y(t). when t = 0.5

At the peak of x(t), y(t), the slope x'(t), y'(t) = 0,

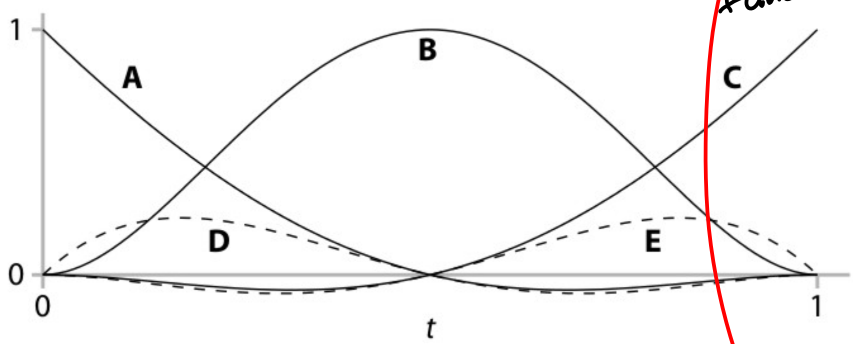
(because

$y'(t) = 12t^2 - 18t + 6 = 6(2t-1)(t-1)$)

and the sign of both x'(t), y'(t) changes when t = 0.5 thus, the cusp appears.

8. When you look at the existing spline code, you see that it is using a sequence of segments, each defined by a quartic (fourth degree) spline with an unfamiliar spline matrix. You plot the basis functions and you see the following curves:

because the basis functions have non-zero tangents at that t.



(a) Which of the five control points (arbitrarily labeled A through E in the plot) will the curve pass through, and for what values of t?

A, B, C $t = 0, \frac{1}{2}, 1$

(b) Which control points affect the tangent to the curve at t = 0? At t = 0.5? At t = 1?

A, D | A, D, E, C | C, E

(c) Does this spline have the convex hull property? How did you tell?

Does not follow convex hull, because there are negative values.

9. Suppose we have an image of a gray elephant, with an alpha matte to delineate foreground from background. The image is stored with pre-multiplied alpha.

a. If we accidentally use the image in a program that expects non-premultiplied alpha, will incorrectly the edges come out too dark, about right, or too light if the background is: → alpha will be multiplied twice.

(a) solid black ($\alpha_B = 1$, $C_B = (0,0,0)$)
(b) solid white ($\alpha_B = 1$, $C_B = (1,1,1)$)
(c) about the same color as the elephant

Explain why by writing down the equations for the correct and incorrect results.

so $C'_A = \alpha^2 C_A$ when the correct value is $C'_A = \alpha C_A$

Hints: using premultiplied alpha C = A **over** B becomes $c'_C = c'_A + (1 - \alpha_A)c'_B$, $\alpha_C = \alpha_A + (1-\alpha_A)\alpha_B$

(a) $C'_C = C'_A = \alpha^2 C_A < \boxed{\alpha C_A}$ (correct answer) → too dark

$\alpha_C = 1$ (because $\alpha_B = 1$)

(b) $C'_C = C'_A + (1-\alpha_A)(1,1,1) = \alpha^2 C_A + (1-\alpha_A)(1,1,1)$

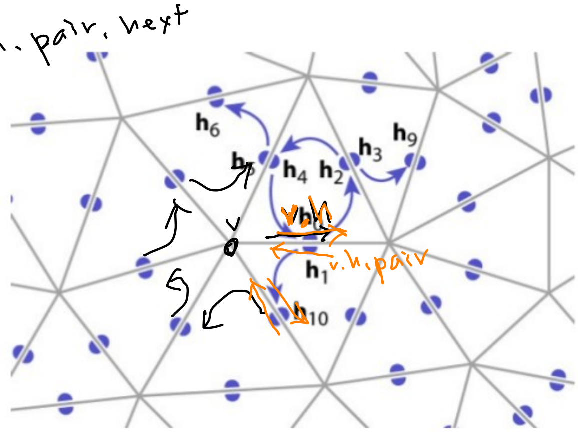
$< \boxed{\alpha C_A + (1-\alpha_A)(1,1,1)}$ (correct answer) → still too dark

(c) too dark no matter what color the elephant is! ($\alpha^2 C_A < \alpha C_A$ for all $0<\alpha<1$

10. EdgesOfVertex is a function that iterates through the list of half edges adjacent to vertex v. Fill in the blanks (???).

```
EdgesOfVertex(v) {
h = v.h;
do {
        h = ???;
    } while (h != v.h);
}
```

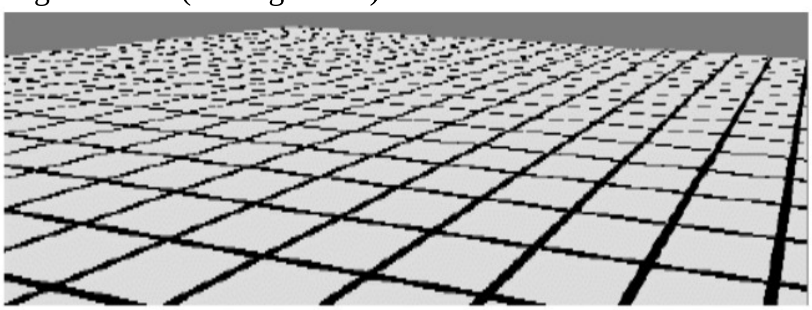
h.pair.next



	pair	next
hedge[0]	1	2
hedge[1]	0	10
hedge[2]	3	4
hedge[3]	2	9
hedge[4]	5	0
hedge[5]	4	6
	⋮	

```
HEdge {
HEdge pair, next;
Vertex v;
Face f;
}
```

11. Briefly describe algorithms to reduce the artifacts caused by the minification (two algorithms) and magnification (one algorithm) of textures.



minification

mipmap, summed area table

preprocess local average of texture values.

magnification

bi-linear interpolation

(converts discrete texture values to continuous ones using a tent-kernel-like reconstruction filter.)

12. Construct a summed area table from the texture below, and explain how you can calculate the average value of the right half of the texture using the summed area table.

1	6	8	3
0	0	3	7
4	7	8	8
5	0	9	9

→

	1	2	3	4
1	1	7	15	18
2	0	7	18	28
3	5	18	37	55
4	10	23	51	78

Sum : $T(4,4) - \cancel{T(4,0)}$ 0

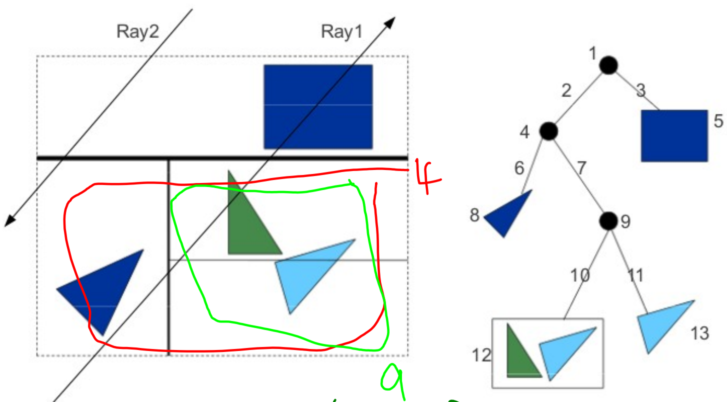
$- T(2,4)$

$+ \cancel{T(2,0)}$ 0

$= 78 - 23 = 55$

$avg = \frac{55}{(4-2)\cdot(4-0)} = \frac{55}{2}$

13. Write down a sequence of numbers in the order that the k-d tree is traversed when checking intersections between a ray and the scene. Each number at an internal node represents the intersection test between the ray and the axis-aligned bounding box corresponding to the node. The numbers at the leaf nodes represents the intersection test between the ray and the primitives. The number at the arrow represents the tree traversal. Omit numbers that correspond to pruned operations (that is not executed). Do not exclude intersection tests that fails; for example, the correct answer is “1” for a ray that doesn't collide with the outer-most bounding box.

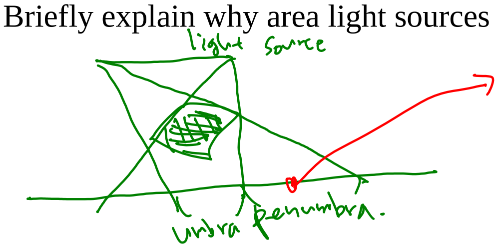


front to back DFS

Ray 1: 1 → 2 → 4 → 6 → 8 → 7 → 9 → 11 → 13 → 10 → 12

Ray 2: 1 → 3 → 5 → 2 → 4 → 7 → 9 → 6 → 8

14. Briefly explain why area light sources generate soft shadows.



at this location, the light source is visible only partially.

15. Briefly explain how a spline curve can be drawn to the screen using OpenGL.

First convert a curve to line segments by uniformly sampling or recursively subdividing the spline curve.
The resulting line segments can be optionally converted to a triangle strip before sending to the graphics pipeline.