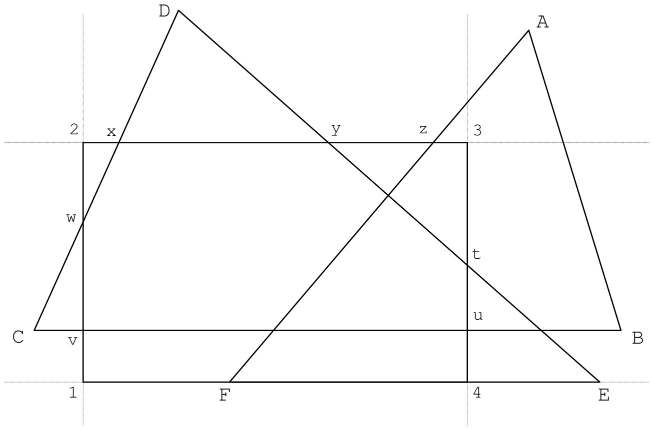
Please read these instructions carefully!

- Answer the test questions in the separate answer sheet.
- You may use the back of all the paper sheets as drafting area (*rascunho*).
- How to: A B C D E
  - Select the answer (A): (● ○ ○ ○ ○)
  - Replace the answer (A) by (C): (⊗ ○ ● ○ ○)
  - Cancel (C) and reactivate (A): (◉ ○ ⊗ ○ ○)
- This test has **??** QUESTIONS (ignoring question zero), each question has a score of 200/**??** points.
- **POINTS LOST** for each wrong answer (in percentage of the question's points):
  $\sum wrong$ $1 = 0\%$ $2 = 11.11\%$ $3 = 22.22\%$ $\geq 4 = 33.33\%$.

Name: ______________________________ Number: ______________

Consider polygon $P = [A, B, C, D, E, F]$, which is going to be clipped by the window $Q = [1, 2, 3, 4]$ using the Sutherland-Hodgeman algorithm. Consider that the clipping order is RIGHT, TOP, BOTTOM, LEFT.



1. How many vertices will the output from stage 1 (clip RIGHT) contain?

   A. 6 **B. 7** C. 8 D. 9

2. Which is the 3rd vertex of the output from stage 3 of the algorithm (clip BOTTOM)?

   A. x B. 4 **C. t** D. y

3. How many edges will the final clipped polygon P' contain?

   A. 12 **B. 10** C. 9 D. 11

4. Choose the correct sequence of vertices in the polygon that comes out of the final stage (Clip LEFT).

   A. ... F z 3 t ... B. ... t u v w ... C. ... 3 t y z ... **D. ... F z 3 u v ...**

Consider that each of the individual line segments of the original polygon P is going to be clipped against the same window Q using the Cohen-Sutherland line clipping algorithm. Imagine also that the order for the bits (from left to right) **is now the following**: TOP, RIGHT, BOTTOM, LEFT.

5. How many line segments would be trivially accepted?

   A. 2 B. 1 **C. none** D. 3

6. How many line segments would be trivially rejected?

   A. 2 B. none **C. 1** D. 3

7. How many line segments would be effectively clipped 2 times?

   A. 2 B. 4 **C. 3** D. none

The FILLAREA (Scanline) algorithm is going to be used to paint the interior of the original polygon P, but now in a window that is large enough so that no clipping wlll occur.

8. How many non empty entries would the Edge Table (Tabela de Arestas) contain?

   A. 1 B. 4 **C. 2** D. 3

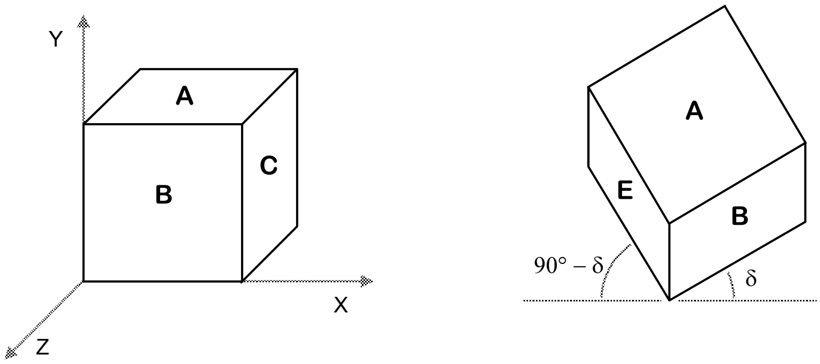
9. what is the total number of edges stored in the Edge Table (Tabela de Areastas)?

   A. 6 **B. 4** C. 5 D. 3

10. Considering that the vertices of polygon P have integral coordinates (located exactly at pixel centers), which of the following options represent the set of vertices that **would not be painted** by the algorithm.

    A. C and F B. C, B, E and F **C. A, B, E and D** D. A and B

In the following figure, faces have been labeled after the projection of a cube for their identification.



11. What is the name of the projection on the left side of the figure, knowing that the object is a cube?

    A. Perspective, center at (0,0,d) **B. Oblique** C. Axonometric D. Perspective, plane at z=-d

12. Which set of values could have been used in the respective projection matrix $\mathbf{M}_?$?

    A. $A = 45°, B = 0°$ **B.** $l = 0.5, \alpha = 45°$ C. $d = 0.5$ D. none of others is correct.

13. The projection of a cube shown on the right side is called bird's eye view or military projection. In this parallel projection the projection plane is parallel to the face labeled as **A**. The angle δ is a parameter of the projection. Using the projection matrix for the type of projection of figure on the left ($\mathbf{M}_?$), what is the expression that computes the projection matrix for the figure on the right?

    A. $\mathbf{M}_? \cdot \mathbf{R_z}(\delta) \cdot \mathbf{R_x}(90°)$ B. $\mathbf{M}_? \cdot R_y(\delta)$ C. $R_y(\delta) \cdot \mathbf{M}_?$ D. $\mathbf{M}_? \cdot \mathbf{R_x}(-90°) \cdot \mathbf{R_y}(\delta)$

14. The projection on the right is sometimes wrongly interpreted as a type of axonometric projection. Which one?

    A. parametric **B. dimetric** C. isometric D. trimetric

15. Consider the `ortho()`, `frustum()` and `perspective()` functions for the definition of a view volume. Choose the sentence that is **false**.

    A. One function is used to desribe a parallelepiped, while the other two define a truncated pyramid.

    B. `frustum()` is more general than `perspective()`.

    C. All functions assume that the camera is located at the origin.

    D. All functions assume that the camera is pointing towards the positive Z-axis.

Consider the Phong illumination model computed by the expression below, for direct illumination (surfaces directly lit by light sources):

$$\mathbf{I} = \mathbf{I}_a\mathbf{K}_a + \mathbf{I}_p[\mathbf{K}_d \cos(\alpha) + \mathbf{K}_s \cos(\phi)^n].$$

16. If we ignore specular reflections, What is responsible for assigning a color to the surface?

    A. $\mathbf{K}_a$ **B.** $\mathbf{K}_a$ **and** $\mathbf{K}_d$ C. $\mathbf{I}_p$ D. $\mathbf{I}_a$

17. The angle α is the angle formed between:

    A. **N** and **H**. **B. N and L.** C. **L** and **V**. D. **R** and **V**.

18. The amount of reflected light given by the specular term is porportional to the n^{th} power of the cosine of the angle formed between:

    A. **V** and **L** **B. R and V** C. **N** and **L** D. **R** and **L**

19. A green object, when lit by a white light, exibits a specular reflection that is yellow. Choose the set of parameters that fit the situation described.

    A. $\mathbf{K}_a = (0.0, 0.3, 0)$, $\mathbf{K}_d = \mathbf{K}_s = (1, 1, 0)$.

    B. $\mathbf{K}_a = \mathbf{K}_d = (0.0, 0.3, 0)$**,** $\mathbf{K}_s = (1, 1, 0)$**.**

    C. $\mathbf{K}_a = (0.3, 0.3, 0)$, $\mathbf{K}_d = \mathbf{K}_s = (0, 1, 0)$.

    D. $\mathbf{K}_a = \mathbf{K}_d = (0.3, 0.3, 0)$, $\mathbf{K}_s = (0, 1, 0)$.

Suppose that a vertex shader implementing the Phong illumination model (in camera coordinates) receives both the View matrix (`mV`) and the current Model matrix (`mM`). It also receives the attributes representing the vertex position (`vPosition`) and its respective normal (`vNormal`).

20. Which of the following would you use to compute vector **L**, for a directional light source, specified in Camera Coordinates through a `uniform vec4` varible named `lightDir`?

    A. `normalize((mV * lightDir).xyz)`

    B. `normalize((mM * mV * lightDir).xyz)`

    C. `normalize((mV * mM * lightDir).xyz)`

    D. **`normalize(lightDir.xyz)`**

21. Which of the following would you use to compute vector **L** for a positional light source, specified in World Coordinates through a `uniform vec4` varible named `lightPos`? Assume that `posC` is the vertex position in Camera Coordinates.

    A. **`normalize((mV * lightPos).xyz - posC)`**

    B. `normalize((mM * mV * lightPos).xyz - posC)`

    C. `normalize((mV * mM * lightPos).xyz - posC)`

    D. `normalize(lightPos.xyz - posC)`

22. Which of the following would you use to compute the vertex position in camera coordinates (`posC`) ?

    A. `(mM * mV * vPosition).xyz`

    B. `vPosition.xyz`

    C. **`(mV * mM * vPosition).xyz`**

    D. `(mV * vPosition).xyz`

23. Complete the sentence: Constant shading is only valid under the assumption that ...

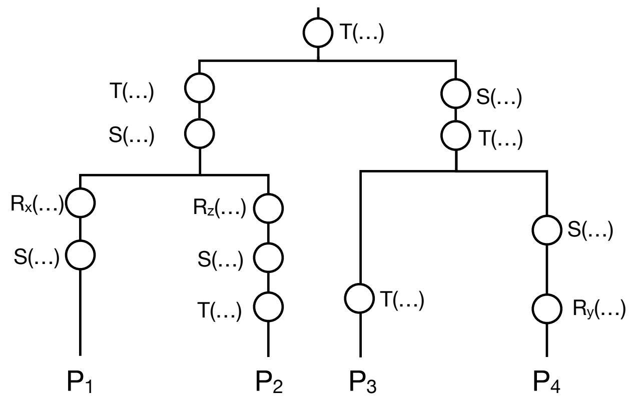
    A. surfaces are flat and both light and viewer have no restrictions on their locations.

    B. light is a point light, the projection is parallel and the faces are flat.

    C. light is directional, the projection is conic (perspective) and surfaces are flat.

    D. **light and viewer are both at infinite distance and that surfces are flat.**

Consider the scene graph on the right, where the parameters for each transformation have been ommited. Each primitive $\mathbf{P}_i$ doesn't change the current transformation matrix.



24. What is the minimum number of `Push()` and `Pop()` **pairs** required to transform the graph into operational drawing code?

    A. 4 B. 6 C. 5 **D. 3**

25. In certain APIs, each transformation node in a scene graph can store multiple transformations. Imagine a system where a transformation node could store any sequence of the form $\mathbf{T} \cdot \mathbf{R} \cdot \mathbf{S}$. How many transformation nodes would you need to represent the same scene?

    A. 11 B. 13 **C. 10** D. 12

26. Imagine that the scene totally fits inside a unit cube centred at the origin. We would like to draw the objects with an image projected on them from above (y-axis points upwards). The projected image would perfectly fit the size of the top and bottom faces of the cube. The primitives already have a set of their own texture coordinates (`attribute vec2 vTexCoords`) assigned to respective positions (`attribute vec4 vPosition`), and the ModelView matrix is accessible through `uniform mat4 mMV`. How would you generate such effect? The vertex shader will assign a `varying vec2 fTexCoords` with the required texture coordinates...

    A. `fTexCoords = vTexCoords + vec2(0.5, 0.5)`

    B. **`fTexCoords = (mMV * vPosition).xz + vec2(0.5, 0.5)`**

    C. `fTexCoords = (mMV * vPosition).xz`

    D. `fTexCoords = vTexCoords`

Boa Sorte!