Nom : Nolet *Prénom :* Charles

17/40

École polytechnique de Montréal
Département de génie informatique et génie logiciel

INF2705: Infographie (Automne 2018)
Contrôle périodique

Notes:

- Toute documentation interdite, calculatrice programmable et ordinateur portatif interdits. Calculatrice non programmable permise.
- Cet examen comprend 4 questions sur 9 pages pour un total de 40 points.

⇒ *Répondez aux questions directement sur le questionnaire.*

Question 1 Transformations dans le pipeline graphique [9 points]

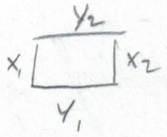
a) [4 points] Nous avons vu que les sommets des primitives sont transformés par le pipeline graphique avant d'être affichés à l'écran. Supposons que les seuls énoncés suivants sont utilisés pour définir le pipeline graphique afin d'afficher dans la partie supérieure de la fenêtre à l'écran :

```
MatricePipeline matrModel, matrVisu, matrProj;

matrModel.LoadIdentity();
matrVisu.LoadIdentity();
//       Ortho( gauche, droite,  bas, haut,  planAvant, planArrière );
matrProj.Ortho( x1, x2,  y1, y2,  0, 10 );

// (Les matrices sont utilisées de façon standard par les nuanceurs)
glUniformMatrix4fv( locmatrProj, 1, GL_FALSE, matrProj );
glUniformMatrix4fv( locmatrVisu, 1, GL_FALSE, matrVisu );
glUniformMatrix4fv( locmatrModel, 1, GL_FALSE, matrModel );

// glViewport( GLint x, GLint y, GLsizei largeur, GLsizei hauteur );
glViewport( 0, h/2, w, h/2 ); // <-- seulement la partie supérieure de l'écran!
```



Dans ce contexte, écrivez les formules qui permettent de convertir un point `(x,y)` en une position `(i,j)` à l'écran, c'est-à-dire de calculer à quel pixel correspond le point. (Ces formules dépendront directement des variables `x`, `y`, `x1`, `x2`, `y1`, `y2`, `w` et `h`.)

(0,0) → $(\frac{w}{2}, \frac{h}{4})$

(x,y)

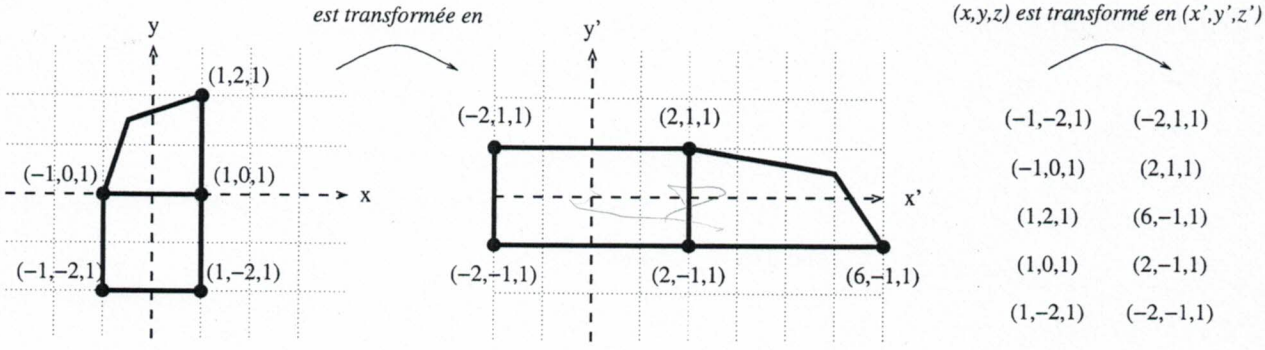
$(\frac{w}{2}+x, \frac{h}{4}+\frac{y}{2})$

i = $\frac{w}{2} + x \quad \frac{x - x_1}{x_2 - x_1} \cdot w$

0

j = $\frac{h}{4} + \frac{y}{2} \quad \frac{y - y_1}{y_2 - y_1} \cdot \frac{h}{2} + \frac{h}{2}$

b) [5 points] Tel que montré ci-dessous, une certaine transformation générale G permet de transformer la figure de gauche en celle de droite.



Sur cette figure, on voit que cette transformation G modifie les coordonnées x et y des sommets sans changer leur coordonnée z. En utilisant des coordonnées homogènes, la transformation générale G peut être exprimée comme le produit de trois transformations élémentaires, $G = T1 \cdot T2 \cdot T3$ et on peut alors appliquer l'opération $\vec{X}' = G \cdot \vec{X}$.

i) Si $T2$ est une transformation élémentaire de rotation, écrivez les noms des deux autres transformations élémentaires $T1$ et $T3$ (sans donner les paramètres) que vous utiliseriez pour construire G.

2 T1 : scaling T2 : *rotation* T3 : translation

ii) Écrivez les deux matrices 4x4 qui correspondent aux transformations $T1$ et $T3$.

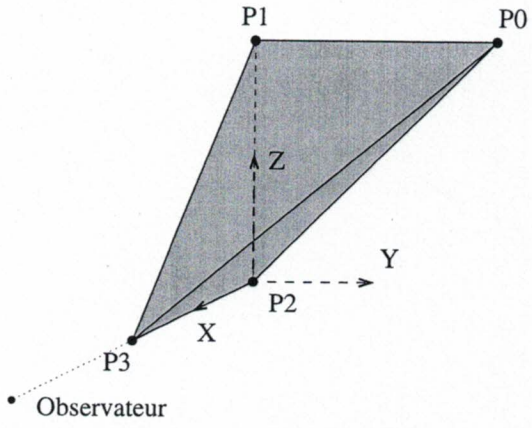
2 $G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Question 2 Notions de base [13 points]

a) [8 points] Considérez la scène illustrée ci-contre qui affiche un tétraèdre composé de quatre triangles construits à partir de ces sommets :

P0 = (0, 1, 1),
P1 = (0, 0, 1),
P2 = (0, 0, 0) et
P3 = (1, 0, 0).

L'observateur est situé au point O = (2, 0, 0) et regarde directement vers le sommet P2. La scène est affichée à l'écran en utilisant une projection orthographique.



i) Par convention, la face avant est celle où l'ordre des sommets est donné dans le sens antihoraire. Si on respecte cette convention, dans quel ordre doivent être donnés les sommets de chacun des quatre triangles composant le tétraèdre afin que chaque face avant soit orientée vers l'extérieur du tétraèdre ?
Écrivez les numéros de sommets (P*i*) dans l'ordre approprié dans la partie gauche du tableau ci-dessous.

ii) Vous savez qu'on doit fournir des vecteurs normaux afin de faire des calculs d'illumination. Si on choisit le modèle d'illumination de Lambert, on a alors besoin d'un seul vecteur par face. Déterminez le vecteur normal de chaque triangle afin qu'il soit lui aussi orienté vers l'extérieur du tétraèdre.
Écrivez chaque vecteur normal (Nx,Ny,Nz) dans la partie centrale du tableau ci-dessous.

iii) Enfin, vous savez aussi qu'on peut demander un affichage sélectif des surfaces selon leur orientation. Si on active cette fonctionnalité (GL_CULL_FACE), seuls les triangles dont l'observateur voit la face avant seront alors affichés et généreront des fragments. De plus, si l'observateur est placé comme indiqué et si une projection orthographique est utilisée pour afficher l'objet en entier en mode « plein » (GL_FILL), seuls certains triangles généreront alors des fragments (... même s'ils ne deviennent pas tous des pixels). Dans ces conditions, écrivez si des fragments seront générés (O/N) pour chaque triangle dans la partie droite du tableau ci-dessous.

	i) 4			**ii)** 2			**iii)** 1.5
Triangle 1 :	P0	P1	P3 ✓	1	0	1 ✓	O
Triangle 2 :	P3	P2	P0 ✓	0	1	-1 ✓	O ✗
Triangle 3 :	P0	P2	P1 ✓	-1	0	0 ✓	N
Triangle 4 :	P1	P2	P3 ✓	0	-1	0 ✓	N

b) [5 points] Trois types distincts de réflexion sont définis dans le modèle proposé par Phong pour calculer l'intensité des réflexions de la lumière en un point : *ambiante*, *diffuse* et *spéculaire*.
Dans ce modèle, identifiez les types de réflexion qui sont influencés par chaque élément ci-dessous.
(Cochez un seul choix)

i) la position de la source lumineuse influence la (les) réflexion(s) ...

- ☐ aucune
- ☐ *ambiante* seulement
- ☐ *diffuse* seulement
- ☐ *spéculaire* seulement
- ☐ *ambiante* et *diffuse*
- ☐ *ambiante* et *spéculaire*
- ☒ *diffuse* et *spéculaire*
- ☐ *ambiante* et *diffuse* et *spéculaire*

ii) la position de l'observateur influence la (les) réflexion(s) ...

- ☐ aucune
- ☐ *ambiante* seulement
- ☐ *diffuse* seulement
- ☒ *spéculaire* seulement
- ☐ *ambiante* et *diffuse*
- ☐ *ambiante* et *spéculaire*
- ☐ *diffuse* et *spéculaire*
- ☐ *ambiante* et *diffuse* et *spéculaire*

iii) la position des autres objets de la scène influence la (les) réflexion(s) ...

- ☒ aucune
- ☐ *ambiante* seulement
- ☐ *diffuse* seulement
- ☐ *spéculaire* seulement
- ☐ *ambiante* et *diffuse*
- ☐ *ambiante* et *spéculaire*
- ☒ *diffuse* et *spéculaire*
- ☐ *ambiante* et *diffuse* et *spéculaire*

iv) le vecteur normal à la surface influence la (les) réflexion(s) ...

- ☐ aucune
- ☐ *ambiante* seulement
- ☐ *diffuse* seulement
- ☒ *spéculaire* seulement
- ☐ *ambiante* et *diffuse*
- ☐ *ambiante* et *spéculaire*
- ☒ *diffuse* et *spéculaire*
- ☐ *ambiante* et *diffuse* et *spéculaire*

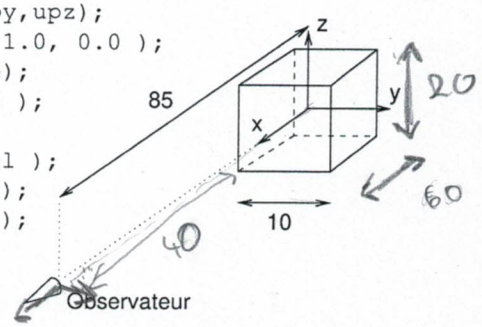
v) les propriétés de matériau influence la (les) réflexion(s) ...

- ☐ aucune
- ☐ *ambiante* seulement
- ☐ *diffuse* seulement
- ☐ *spéculaire* seulement
- ☐ *ambiante* et *diffuse*
- ☒ *ambiante* et *spéculaire*
- ☐ *diffuse* et *spéculaire*
- ☒ *ambiante* et *diffuse* et *spéculaire*

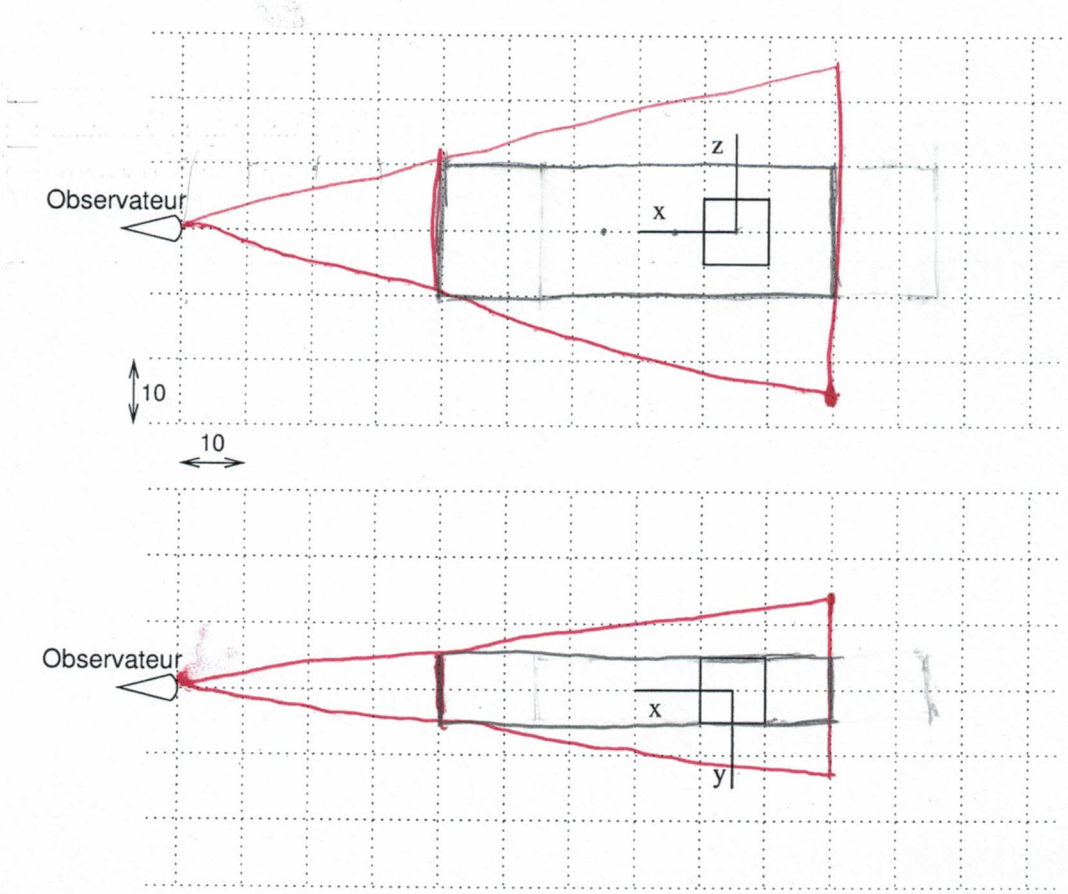
Question 3 Visualisation 3D [11 points]

Une application OpenGL affiche un cube centré à l'origine, dont chaque arête a une longueur de 10 unités. Cette application utilise `Frustum()` pour générer une matrice de projection et `LookAt()` pour positionner l'observateur directement sur l'axe des ~~x~~ z à une distance de 85 unités du centre du cube (c'est-à-dire à 80 unités de la face avant du cube). La clôture de la fenêtre à l'écran est rectangulaire de 400 x 200 pixels. Prenez bien note que le *vecteur up* pointe dans la *direction de l'axe des Y* !

```
MatricePipeline matrModel, matrVisu, matrProj;
matrModel.LoadIdentity();
// LookAt(obsx,obsy,obsz,ptViséx,ptViséy,ptViséz,upx,upy,upz);
matrVisu.LookAt( 85.0, 0.0, 0.0,  0.0, 0.0, 0.0,  0.0, 1.0, 0.0 );
// Frustum(Gauche,Droite,Bas,Haut,PlanAvant,PlanArrière);
matrProj.Frustum( -10.0, 10.0,  -5.0, 5.0,  40.0, 100.0 );
// Passer les matrices aux nuanceurs
glUniformMatrix4fv( locmatrModel, 1, GL_FALSE, matrModel );
glUniformMatrix4fv( locmatrVisu, 1, GL_FALSE, matrVisu );
glUniformMatrix4fv( locmatrProj, 1, GL_FALSE, matrProj );
glViewport( 0, 0, 400, 200 );
```

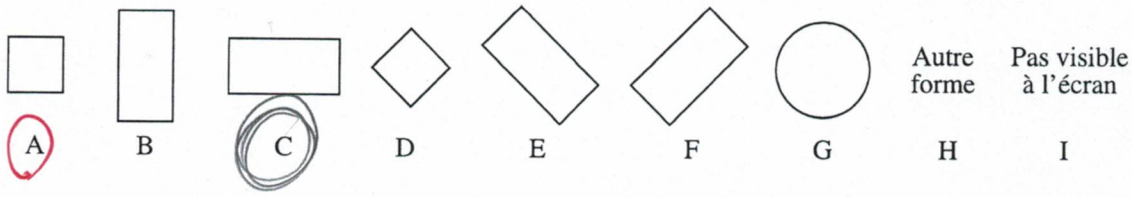


a) [4 points] Tracez, en vue de plan et en vue de côté, le volume de visualisation spécifié par l'appel à `Frustum()`. Assurez-vous surtout de bien montrer les coins du volume de visualisation.



b) **[1 point]** Considérant les projection et clôture utilisées, comment la face avant du cube apparaît-elle à l'écran ?

0



c) **[2 points]** Quelles sont les dimensions à l'écran (en pixels) de la face avant du cube ?

0

100 100

Dimensions = 20 x 10 pixels

d) **[2 points]** En respectant le rapport d'aspect actuel et sans modifier PlanAvant ou PlanArrière, quelles modifications doit-on apporter à Gauche, Droite, Bas et Haut afin de cadrer la face avant du cube, c'est-à-dire afin qu'on continue à voir la face au complet tout en occupant le maximum de pixels à l'écran ?

0

-5 5 -2.5 2.5

Gauche = -200 Droite = 200 Bas = -100 Haut = 100

e) **[1 point]** Avec les modifications de la sous-question précédente, quelles seront alors les dimensions à l'écran (en pixels) de la face avant du cube ?

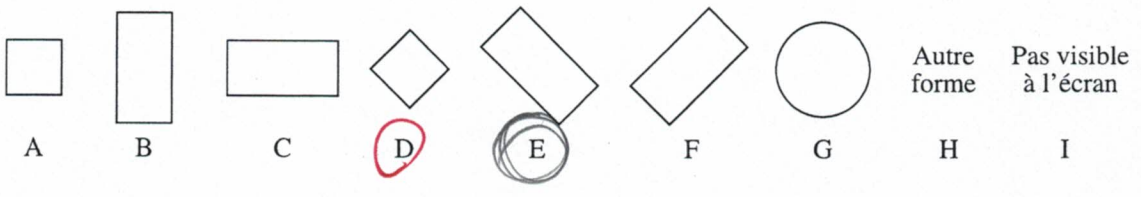
0,5

200

Dimensions = ~~400~~ x 200 pixels

f) **[1 point]** En utilisant à nouveau les valeurs originales de Gauche, Droite, Bas, Haut, mais si on changeait le vecteur up pour avoir `(upx,upy,upz) = (0.0, 1.0, 1.0)`, comment la face avant du cube apparaîtrait-elle alors à l'écran ?

0



Question 4 Fragments [7 points]

Toutes les sous-questions qui suivent présentent différentes situations pour lesquelles on souhaite déterminer le résultat d'un « test unitaire » pour tester l'effet de certains énoncés OpenGL. Chaque sous-question est indépendante des autres.

Pour chaque test, on vous donne les valeurs des attributs du fragment courant et les valeurs présentes dans les différents tampons (*profondeur, couleur, stencil*) et, selon les énoncés OpenGL, on vous demande d'écrire les valeurs subséquentes qui seront présentes dans les différents tampons.

a)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.5	(0.8, 0.8, 0.8, 0.3)
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)

```
glDisable( GL_STENCIL_TEST );
glDisable( GL_DEPTH_TEST ); // Désactiver ce test fait en sorte que le
                            // tampon de profondeur ne sera pas modifié
glDisable( GL_BLEND );
```

0-5

0.7

- valeurs subséquentes dans les tampons	0,5	(0,8, 0.8, 0.8, 0.3)

b)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.5	(0.8, 0.8, 0.8, 0.3)
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)

```
glDisable( GL_STENCIL_TEST );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_GREATER ); // " > ", l'inverse de la valeur par défaut
glDisable( GL_BLEND );
```

1

- valeurs subséquentes dans les tampons	0,7	(0.4, 0.4, 0.4, 0.4)

Note :
```
void glStencilFunc( GLenum func, GLint ref, GLuint mask );
void glStencilOp( GLenum sfail, GLenum zfail, GLenum pass );
```

c)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.5	(0.8, 0.8, 0.8, 0.3)	-
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	**2**

```
glEnable( GL_STENCIL_TEST );
glStencilFunc( GL_GEQUAL, 1, 3 ); // " >= "
glStencilOp( GL_KEEP, GL_INCR, GL_REPLACE );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_LEQUAL ); // " <= "
glDisable( GL_BLEND );
```

greater

fail donc on arrête là

0.7 2

- valeurs subséquentes dans les tampons	0,5	(0.4, 0.4, 0.4, 0.4)	3

d)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.5	(0.8, 0.8, 0.8, 0.3)	-
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	**2**

```
glEnable( GL_STENCIL_TEST );
glStencilFunc( GL_LESS, 1, 3 ); // " < "
glStencilOp( GL_KEEP, GL_INCR, GL_REPLACE );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_GREATER ); // " > ", l'inverse de la valeur par défaut
glDisable( GL_BLEND );
```

3

- valeurs subséquentes dans les tampons	0,7	(0.4, 0.4, 0.4, 0.4)	2

e)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.5	(0.8, 0.8, 0.8, 0.3)	-
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	**2**

```
glEnable( GL_STENCIL_TEST );
glStencilFunc( GL_GREATER, 1, 3 ); // " > "
glStencilOp( GL_KEEP, GL_INCR, GL_REPLACE );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_LESS ); // " < "
glDisable( GL_BLEND );
```

fail donc on arrête là

0,7 0.4 0.4 0.4 0.4 2

- valeurs subséquentes dans les tampons	0,5	(0.8, 0.8, 0.8, 0.3)	3

Note : `void glBlendFunc( GLenum sfactor, GLenum dfactor );`

f)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.5	(0.8, 0.8, 0.8, 0.3)
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)

```
glDisable( GL_STENCIL_TEST );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_ALWAYS ); // " toujours "
glEnable( GL_BLEND );
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA ); // " SrcA, 1-SrcA "
```

0.5

0.3 · 0.8 = 0,24
0.7 · 0.4 = 0.28

- valeurs subséquentes dans les tampons	0.5	(0.52, 0.52, 0.52, 0.7) 0.37

g)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.5	(0.8, 0.8, 0.8, 0.3)
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)

```
glDisable( GL_STENCIL_TEST );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_LESS ); // " < "
glEnable( GL_BLEND );
glBlendFunc( GL_ONE, GL_ONE_MINUS_SRC_ALPHA ); // " 1, 1-SrcA "
```

0.5

0.5 < 0.7 ✓

0.8 · 1 = 0.8
0.4 · 0.7 = 0.16

- valeurs subséquentes dans les tampons	0.5	(1, 1, 1, 0.7) 0.58

Cet examen comprend 4 questions sur 9 pages pour un total de 40 points.
Benoît Ozell