Bilan

		Note finale(/20)	Pointeurs (/5)	STL (/5)	Heritage(/5)
ALLAIN	Clément	19	4	4.5	4
BAKONG EPOUNÉ	Killian	15	3	4.5	3
BEL KHAYAT ZOUGGARI	Yassine	14.5	3	4.5	2
BERTRAND	Romain	14	2	3.5	4
BIRAC	Nicolas	16.5	4	4	4.5
BOUGHDIRI	Ahmed	15.5	5	5	3.5
BRAMI	Victoria	12.5	1	4.5	1
BRISINGER	Maxime	15.5	4	4.5	3.5
CANCES	Adrien	17.5	3.5	4.5	4
CHAKIR	Adel	12.5	3	4	1
CHAMBON	Loick	16.5	4	4.5	4.5
CHANCEREL	Pia	15	4	4	1
COSSON	Margot	17.5	4.5	4.5	4
DESHORS	Charles	15.5	3	4.5	3.5
DESJARDINS	Antoine	13	3	4.5	2
DURIF	Anne	11.5	1.5	3.5	1.5
FATOUX	Lambert	14.5	3	4	4
FAUDUET	Alex	17.5	4.5	4	5
FERNANDEZ	Raphaël	15	3.5	4.5	2
FRANCOIS	Quentin	15.5	4	3.5	4
GAINON	Louise	15.5	3.5	4	4
GINOULHAC	Raphaël	18.5	4.5	5	5
HÉMADOU	Louis	14.5	2	4	4
LAINÉE	Martin	7.5	1.5	4.5	1
LANDAIS	Jean	15	3.5	4	3.5
LASRY	Raphaël	17	4.5	4.5	3.5
LÊ	Paul-Vinh	17	5	3.5	3.5
LE GRELLE	Hugues	16.5	5	4.5	3
LESUEUR	Louis	16.5	5	4.5	4
LOISON	Virginie	13	2	3	2.5
MEDINA	François	14	3.5	4	1.5
MOREL	Sébastien	15.5	4.5	3.5	3
OUHAÏCHI	Firas	12.5	3.5	4	4
PAN	Chao	16.5	4.5	4	4
PION	Aurélien	15.5	4	4	4
POLETTE	Nadège	19.5	5	5	3.5
POLI	Maxime	20	5	4.5	4.5
POUCIN	Florentin	14	3	4	3
QUILY	Ludovic	15.5	4	3.5	4.5
RIOU	Auriane	15.5	4.5	4	2
RUGET	Simon	16.5	4	3	4
SANCHEZ CRUZ	Jenny		1.5	2.5	
SCHLEGEL	Nicolas	16	3	4	3.5
SOUSSI	Nada	15.5	4.5	4	1.5
STÄMPFLI	Erwan	13.5	4	4	4
TRINH	Robin	10	2	1	2
VAN DEN BERGH	Candice	18	4	5	4.5

VENARD	Paul-Louis	13	4	4	2.5
VINCENT	Théo	14.5	3.5	4.5	2.5
VONGPASEUT	Clarine	15.5	2	4.5	4

Data(/5)	ProgDyn(/5)	QuadTree(/5)	SetHash(/5)
5.5	4.5	5.5	5
2.5	4.5	4	4.5
4	3	4	4.5
4	4	3	4
4.5	4.5	3.5	3.5
5.5	4	0	4.5
4	4	4	3
3.5	4	4.5	3.5
4.5	4.5	5.5	4.5
3.5	3.5	2.5	4
4	4	4.5	3.5
4.5	4	4.5	4.5
4.5	4	4.5	5
4	3.5	4	4.5
4	4	1.5	3.5
4	3.5	3.5	2.5
4	4	3	3.5
5	4	4	4
4	3.5	4.5	4.5
4	4	4.5	3.5
3.5	4.5	3.5	4.5
4.5	4.5	4	5
3	4	4.5	4
0	3.5	1.5	1.5
5	3.5	2	5
4	4	4.5	4.5
5	4	4	4.5
4	3.5	4.5	4.5
2.5	3.5	4.5	4.5
2	4	4	5
3.5	2.5	4.5	5
4.5	3.5	4.5	4
2.5	3	4	1
5	3.5	4	3.5
4	3.5	4	3.5
5.5	4.5	5.5	5
5.5	5	5.5	5
3.5	4	4	3
4.5	4	4	3
4.5	4.5	4	4
5	4.5	4	4
4.5	5	3.5	4.5
5	3.5	4.5	4.5
2	4	3.5	2
4	4	1.5	3
4.5	4.5	4.5	4.5

3.5	3	4	2
3.5	3.5	3.5	4.5
4	4	4.5	4

Pointeurs (/5)

ALLAIN	Clément	4	Desole, ca ne compile pas avec gcc 7.4 (nombreuses erreurs) et je n'ai pas le temps d'aller reparer une telle monstruosite (= code tres complexe et non commente). Tu n'as fait que des matrices statiques ?
BAKONG EPOUNÉ	Killian	3	compteur++ et compteur-- ne peuvent pas marcher car ++ et – s'appliquent a compteur et non a compteur. Il faut des parentheses. Ceci corrige, il ne faut pas oublier de faire le delete compteur lors de la destruction quand compteur devient 0. Les get/set auraient du etre operator(). Operator* devrait prendre des const matrice&, ce qui ne marcherait pas ici car get_nb ne sont pas const.
BEL KHAYAT ZOUGGARI	Yassine	3	On ne doit pas faire delete A deux fois, c'est interdit et provoque un crash. Les get_data et get_c sont inutiles et potentiellement nuisibles. L'operator= est bancal, il ne devrait rien copier, mais simplement jouer sur les pointeurs.
BERTRAND	Romain	2	Un crash dans le programme car il manque compteur=A.compteur dans le constructeur par copie. Petite fuite memoire car compteur n'a jamais de delete. cout << "" n'a aucun effet. Ne pas mettre les champs publics.
BIRAC	Nicolas	4	Bien dans l'ensemble mais operator= a un probleme : il devrait faire comme le destructeur suivi du constructeur par copie. Il faut donc faire attention au compteur, ne pas reallouer de memoire, etc.
BOUGHDIRI	Ahmed	5	TB
BRAMI	Victoria	1	Le programme ne compile pas a cause de confusions entre pointeurs et references. De plus, attention, compteur est un pointeur donc compteur-1 aussi, ce qui n'est pas du tout pareil que *compteur-1
BRISINGER	Maxime	4	Ne pas faire cout de *compteur apres delete compteur. Il aurait mieux valu operator() plutot que lecture/ecriture. Operator= doit commencer par faire comme le destructeur. Ne pas faire de new dans la definition de compteur dans le .h.
CANCES	Adrien	3.5	Le constructeur par copie est cense ne faire aucune allocation, simplement partager le champ tab. Ca a aussi une consequence dans le destructeur, qui ne doit pas liberer tab quand d'autres l'utillisent. Ta matrice A2 n'est pas initialisee.
CHAKIR	Adel	3	Il y a un pb dans la linearisation des indices (i,j), d'ou des erreurs memoire. Mettre const a Affichematrice, qui d'ailleurs aurait mieux fait d'etre un operator<< avec ostream.

CHAMBON	Loick	4	Il manque delete compteur quand *compteur passe a 0. Les fonctions get/set sont inutiles, avantageusement remplacees par operator(). Mettre affichage() const. Le programme echoue sur un assert lorsque m et n sont differents.
CHANCEREL	Pia	4	Il manque un operator=, qui est pourtant utilise dans A=D. Du coup, il ne se comporte pas bien avec le compteur. Il aurait ete preferable d'avoir operator() plutot que get/set.
COSSON	Margot	4.5	Ton operator= ne traite pas les matrices de la meme facon et est potentiellement dangereux. Il devrait fonctionner avec le compteur et ne pas faire d'allocation. Sinon, le reste est tres bien. Attention toutefois de ne pas surcharger de commentaires, ca nuit a la lecture du code.
DESHORS	Charles	3	Le delete[] tab ne doit s'appliquer que lorsque le compteur est a 0. Il aurait ete bien de mettre des assert dans operator(). Affiche() devrait etre const.
DESJARDINS	Antoine	3	Le destructeur ne doit faire delete que lorsque compteur passe a 0. operator= ne devrait pas allouer de memoire. Matrice::operator* devrait etre const.
DURIF	Anne	1.5	On ne peut pas faire *compteur=1 tant que compteur ne pointe pas sur une variable. Pour cela, il faut faire un new. Le destructeur ne doit desallouer que lorsque le compteur tombe a zero. Le constructeur par copie ne doit pas reallouer de memoire, de meme que operator=.
FATOUX	Lambert	3	Dans ton cas, operator= est indispensable car celui par defaut ne gere pas comme il faut le compteur. Il manque delete compteur dans le destructeur. i*m+j est incorrect car m est la borne de i, pas de j.
FAUDUET	Alex	4.5	Bien, mais ca aurait ete bien de faire des matrices aleatoires et de tester sur un peu plus que simplement des matrices 2x2.
FERNANDEZ	Raphaël	3.5	Il faut cacher les champs en private. Compteur n'est pas un tableau, donc ne pas utiliser delete[]. Le main fait des new qui ne sont jamais desalloues. Il aurait ete bien de faire des operator() pour acceder aux coefficients.

FRANCOIS	Quentin	4	operator() const ne doit pas retourner double&, car il permet de modifier les coefficients d'une matrice constante. L'operator= est deficient, il doit faire comme le destructeur et la recopie. Faire compteur=A.compteur puis compteur[0]=A.compteur[0] est redondant. get/set sont inutiles.
GAINON	Louise	3.5	Le constructeur par copie fait (*compteur)++ alors que compteur n'est pas initialise, d'ou un crash du programme ! Ne pas mettre compteur public. Ajouter const pour operator*.
GINOULHAC	Raphaël	4.5	operator= est defectueux, mais non utilise par le test. Bien mettre const les operateurs et methodes qui le meritent. Definir operator<< plutot que affiche.
HÉMADOU	Louis	2	Une erreur dans operator*, get(i,k)*get(k,j) au lieu de M.get(k,j). operator= n'est pas correct, plusieurs erreurs. Le constructeur par copie fait ++*compteur avant que compteur n'ait ete initialise ! De plus, aucune allocation memoire ne devrait avoir lieu dans ce constructeur.
LAINÉE	Martin	1.5	Le constructeur de Matrice ne peut pas faire *compteur=1 sans avoir initialise compteur, ce qui se fait par un new. Le destructeur est correct, tout comme le constructeur par copie. Faire des operateurs tant que possible. Mettre const les methodes qui le peuvent.
LANDAIS	Jean	3.5	Matrice::operator= est defectueux, car il detruit systematiquement le tableau, sans tenir compte d'autres matrices qui pourraient le partager. Il aurait mieux valu operator() que get_coeff et set_coeff. Bien mettre const les methodes qui le meritent. Utiliser des assert dans l'acces aux coefficients.
LASRY	Raphaël	4.5	operator= ne fonctionne pas comme il faut: commencer par se detacher des donnees actuelles (decrement compteur), puis se connecter aux nouvelles. Ne pas sur-commenter, ca nuit a la lisibilite du code. En particulier, ne pas faire des lignes trop longues.
LÊ	Paul-Vinh	5	TB. Il aurait ete bien de se debarrasser de set et de remplacer par operator().
LE GRELLE	Hugues	5	TB. Respecter les majuscules/minuscules dans les noms de fichiers, include et CmakeLists.txt
LESUEUR	Louis	5	TB. Ne pas melanger tabulations et espaces pour indenter, ca ne s'affiche correctement que si le tab equivaut a x espaces, parfois x=4 ou x=8 suivant les editeurs.

LOISON	Virginie	2	Le principe de la copie superficielle n'est pas compris, le compteur ne sert a rien dans ton implementation. Confusion entre ligne et colonne dans operator().
MEDINA	François	3.5	Il y a en fait des fuites memoire a cause du compteur--. Cette instruction decremente le pointeur, donc le fait pointer sur une case memoire interdite. Il ne faut pas confondre compteur (pointeur) et *compteur (variable de type int). D'autre part, il aurait ete bien de mettre const les methodes comme *affiche et operator*.
MOREL	Sébastien	4.5	get/set auraient pu etre remplaces par operator(), et en plus faire un assert. Operator* devrait etre const.
OUHAÏCHI	Firas	3.5	Un crash car *compteur++ increment compteur et non *compteur. Dommage, car le reste est bien, y compris operator=. get/set devraient faire un assert et etre remplaces par operator().
PAN	Chao	4.5	Passer des const Matrice& aux fonctions (ne pas oublier le const).
PION	Aurélien	4	Petite fuite memoire car il manque delete counter. L'operator= n'est pas defini, or tu l'utilises en faisant C=M*M. Celui par defaut va oublier d'incrementer le compteur, ce qui provoque une erreur par la suite.
POLETTE	Nadège	5	TB. Mettre const les methodes qui le peuvent. Remplacer get/set par operator().
POLI	Maxime	5	Parfait ! Seul bemol, operator<< aurait ete mieux qu'une methode affiche.
POUCIN	Florentin	3	Le constructeur par copie ne doit pas allouer tab, mais juste pointer sur celui de l'objet qu'il copie. Le destructeur ne doit desallouer que quand compteur atteint 0.
QUILY	Ludovic	4	Le principe est compris, mais il aurait fallu tester sur des matrices plus que 2x2. Rendre const operator+ et operator*.
RIOU	Auriane	4.5	Ton operator= est incorrect, mais il n'est pas utilise par ton main. Le principe est compris.
RUGET	Simon	4	A chaque fois que tu fais --*compteur, il faut verifier si on ne tombe pas a 0, auquel cas il faut desallouer. C'est le cas dans operator= et operator*=
SANCHEZ CRUZ	Jenny	1.5	C'est tres incomplet, il manque la copie et operator*. Il y a un bug grave dans le constructeur: int var=1; compteur=&var fait pointer compteur sur une variable qui n'existera plus en sortie du constructeur. Il faut faire un new pour obtenir une variable persistante. Ne pas creer un projet Qt a partir de QtCreator, ce n'est pas la bonne facon de faire.

SCHLEGEL	Nicolas	3	Faire tab= new double[] suivi de tab= fait une fuite memoire (le tableau alloue). Mettre des assert dans l'acces aux elements, ecrire des operator(), mettre operator* const. Utiliser delete sans [] pour compteur qui n'est pas un tableau.
SOUSSI	Nada	4.5	Petit defaut dans operator= qui peut omettre de desallouer quand c'est le dernier pointeur. Sinon, c'est tres bien.
STÄMPFLI	Erwan	4	Confusion entre lignes et colonnes dans operator(), supprimer les assert n'etait pas la bonne reponse. Supprimer le compteur dans le destructeur n'est pas inutile, toute fuite memoire doit etre prohibee.
TRINH	Robin	2	Plusieurs pbs: il n'y a pas de constructeur par copie (ne pas confondre avec le rvalue reference Matrice&&). Faire *count++ incremente count et non *count. Le destructeur risque de faire un double delete[]. De plus, il decrement deux fois le compteur.
VAN DEN BERGH	Candice	4	Bon ensemble, sauf ton operator*= qui est dangereux: si la matrice partageait des coefficients, ceux-ci sont detruits sans avertissement des partenaires. Il faut agir comme pour le destructeur, en prenant en compte le compteur.
VENARD	Paul-Louis	4	Les methodes lecure/ecriture ne sont pas tres elegantes. Le constructeur sans argument est incorrect, car tab vaut n'importe quoi. Il faudrait definir un operator= puisque ton programme l'utilise.
VINCENT	Théo	3.5	Le programme plante assez tot a cause de + +*compteur, alors que compteur n'est pas initialise. Modulo cela, le principe est correct. Le constructeur sans argument devrait etre supprime, il est bancal (compteur non initialise). Utiliser operator() plutot que get/set.
VONGPASEUT	Clarine	2	Il ne faut **jamais** appeler explicitement un destructeur. ProduitMatrice n'est pas tres elegant, remplacer par operator* et rendre const.

		STL (/5)	
ALLAIN	Clément	4.5	Tu pouvais faire un peu mieux que lister tous les caracteres pour le nom d'un eleve. Bien sinon.
BAKONG EPOUNÉ	Killian	4.5	Il est assez maladroit de faire une liste des lettres, alors qu'elles ont des codes ASCII consecutifs. Il vaut mieux mettre les champs ptr en private des foncteurs.
BEL KHAYAT ZOUGGARI	Yassine	4.5	La condition i<f(rand()) dans le for n'est pas tres avise : a chaque iteration, la fonction rand est reappelee, donc il y a peu de chances d'avoir des noms longs. Mettre const les methodes qui ne modifient pas Eleve.
BERTRAND	Romain	3.5	Faire int* ptr=new int suivi de ptr= fait que le resultat du new est une fuite memoire. De plus, ne pas initialiser directement dans la declaration, c'est le constructeur qui doit le faire. Il n'y a pas de majuscule en initiale du nom, de plus faire la liste des lettres est inutiles, elles sont consecutives. Le 1+rand() %26 fait que le a (lettre 0) n'apparait jamais et que tu peux sortir du tableau (indice 26).
BIRAC	Nicolas	4	Mettre en dur les codes ASCII des caracteres a et A n'est pas ideal. La fonction affiche doit prendre le vecteur par reference pour eviter une copie. La note de 20 est inatteignable pour un Eleve aleatoire. Mettre const les methodes de Eleve. Tu n'as pas compris le bon usage de srand.
BOUGHDIRI	Ahmed	5	Il n'est pas ideal de mettre toutes les lettres dans un string, il vaut mieux utiliser la consecutivite des codes ASCII. Bien par ailleurs.
BRAMI	Victoria	4.5	La fonction affiche devrait prendre le vecteur par reference pour eviter une copie. Coder dans un string la liste des lettres n'est pas optimal.
BRISINGER	Maxime	4.5	Les operator() des foncteurs devraient prendre des const Eleve&, ce serait plus efficace. La liste explicite des caracteres est inefficace, utiliser leur consecutivite.
CANCES	Adrien	4.5	La liste explictie des lettres est inutile, elles ont des codes ASCII consecutifs. Les operator() des foncteurs devraient prendre des const Eleve& pour etre plus efficaces. La comparaison lexicographique des string est correcte, mais l'operateur< existe deja et le fait tout aussi bien.
CHAKIR	Adel	4	La liste des caracteres aurait pu etre evitee en utilisant leur consecutivite. Les operator() devraient prendre des const Eleve& plutot. L'affichage aurait pu etre plus clair, avec a chaque fois le nom et la note.

CHAMBON	Loick	4.5	La methode valeur_pointeur est assez maladroite, puisque la variable pointee est toujours disponible par le createur du foncteur. Les operator() devraient prendre des const Eleve&
CHANCEREL	Pia	4	La fonction template print dans eleves.cpp est malvenue: donner un nom de classe comme parametre template est meme peut-etre interdit. De plus, comme on ne fait pas #include d'un cpp, on ne peut pas l'utiliser. Mettre const les methodes de Eleve qui peuvent l'etre. Passer les champs ptr en private. Ne pas liste explicitement les caracteres admissibles, toutes les lettre le sont.
COSSON	Margot	4.5	Il aurait ete facile de ne pas lister explicitement toutes les lettres en utilisant la consecutivite des codes ASCII. Les foncteurs auraient pu prendre des const Eleve& au lieu de Eleve&. Bon ensemble sinon. Attention de ne pas surcharger le code de commentaires, surtout quand cela fait des lignes de code vraiment longues.
DESHORS	Charles	4.5	Les operator() auraient pu prendre des const Eleve&. Pas de liste explicite des caracteres.
DESJARDINS	Antoine	4.5	Il aurait ete bien de mettre const les methode de Eleve et de passer des const Eleve& aux foncteurs. La note ne peut pas depasser 19 si tu fais %20. ne pas coder en dur les codes ASCII de la lettre a.
DURIF	Anne	3.5	Il n'est pas tres judicieux d'appeler un objet avec le meme nom que sa classe: CompareNom et CompareNote. Ton affichage est minimal et n'utilise qu'un foncteur. Il faut rendre const les methodes de Eleve et passer des const& a operator(). Mettre les ptr en private. Ecrire if(cond) return true; else return false est maladroit, return cond fait directement la meme chose.
FATOUX	Lambert	4	Ce n'est pas une bonne idee d'avoir la fonction print template nommant son parametre du nom de la classe Eleve. Il aurait ete bien de passer des const Eleve& partout ou c'est possible. Ne pas mettre ptr dans la partie public. La liste des lettres est inutile, il y avait mieux a faire.
FAUDUET	Alex	4	Attention la relation de comparaison attendue par sort doit etre stricte, < et non <=. Il aurait ete bien de mettre const les methodes de Eleve et de passer des const Eleve& a operator(). Ne pas coder la valeur numerique de la lettre a.

FERNANDEZ	Raphaël	4.5	Pas besoin de liste explicite des lettres, leur consecutivite suffit pour faire plus simple. Passer des const Eleve& a operator(). Il n'est pas tres judicieux de definir random dans main.cpp, alors qu'il n'est utilise que dans eleves.cpp. L'ordre lexicographique des string est simplement l'operator <.
FRANCOIS	Quentin	3.5	J'ai un crash du programme a cause du delete ptr dans les destructeurs des foncteurs. C'est une erreur grave, car ptr n'a pas ete obtenu par new. La methode Get_Compteur est maladroite et inutile, on a toujours les compteurs directement par leur variable.
GAINON	Louise	4	Il aurait fallu cacher le ptr dans la partie privee du foncteur. Attention de reinitialiser le compteur a 0 entre les deux tris. Ne pas mettre en dur les valeurs numeriques du caractere a.
GINOULHAC	Raphaël	5	TB, il aurait juste ete parfait de mettre les ptr dans la partie privee.
HÉMADOU	Louis	4	Passer le vector par reference a affiche_prom pour eviter une copie. Mettre const les methodes de Eleve. Cacher ptr dans la partie privee du foncteur. Passer des const Eleve& a operator(). Ne pas lister toutes les lettres possibles, utiliser leur consecutivite.
LAINÉE	Martin	4.5	Rendre const les methodes de Eleve et passer des const& a operator(). Ne pas lister explicitement les caracteres.
LANDAIS	Jean	4	Pourquoi compareNom::operator() ne compare-t-il que la premiere lettre du nom? Passer des const Eleve& a cet operateur evite des copies inutiles. Bien mettre const les methodes de lecture des champs de Eleve. Ne pas mettre d'instruction en #ifndef et #endif ne sert a rien. Il faut enlever ca, le #pragma once le remplace avantageusement.
LASRY	Raphaël	4.5	Mettre const les methodes lire_xxx de Eleves aurait permis de passer des const Eleve& a operator(). Le nom Eleves n'est pas tres approprie, il aurait mieux value Eleve au singulier. La liste explicite des lettres aurait pu etre evitee.
LÊ	Paul-Vinh	3.5	J'ai un crash a cause du delete ptr dans le destructeur du foncteur, alors que ptr n'a pas ete obtenu par un new. Passer un vector& a affiche_tableau. Ne pas coder en dur les codes ASCII. Mettre const les methodes de Eleve.

LE GRELLE	Hugues	4.5	Rendre const les methodes de Eleve aurait permis de passer des const Eleve& a operator(). La methode affichecompteur n'est pas franchement utile, on a toujours la variable par ailleurs.
LESUEUR	Louis	4.5	Il aurait ete preferable de rendre const les methodes de Eleve et de passer des const Eleve& partout. Ne pas mettre en dur les codes ASCII.
LOISON	Virginie	3	Tu n'as pas mis de compteur dans les foncteurs ? C'est dommage, ca illustrait l'utilite d'avoir un foncteur.
MEDINA	François	4	Tu as utilise deux fois CompareNom, et pas CompareNote. Ne pas utiliser en dur les codes ASCII de 'a'. Faire %21 pour avoir une note jusqu'a 20. Faire %26 pour ne pas exclure la lettre z. Passer le vecteur par reference a affiche_classe pour eviter une copie.
MOREL	Sébastien	3.5	L'affichage du programme laisse un peu perplexe. Il est etrange que printnom et printnote soient template, avec qui plus est un parametre template du nom de la classe Eleves... Ne pas mettre ptr public dans les foncteurs, utiliser un constructeur. Le constructeur par copie de Eleve est inutile. Il est inutile d'acceder au compteur via la foncteur, il est deja disponible en variable.
OUHAÏCHI	Firas	4	Il aurait ete mieux de passer const Eleve& a operator(). La methode affiche_compteur est inutile, on a deja la variable pointee. Ne pas lister explicitement la liste des lettres.
PAN	Chao	4	Mettre en private le champ ptr et utiliser un constructeur pour l'initialiser. Mettre const les accesseurs en lecture de Eleve, et passer des const& a operator(). Un seul tri est effectue par ton programme. L'operator() doit se comporter comme une comparaison stricte pour que sort fonctionne correctement.
PION	Aurélien	4	Attention, compteur est partage par cmpnom et cmpnote et non remis a 0 entre les deux tris. Le fichier eleve.cpp etant vide est inutile. Rendre private le ptr du foncteur et l'initialiser par un constructeur.
POLETTE	Nadège	5	Parfait !
POLI	Maxime	4.5	Il vaut mieux cacher le ptr dans la partie privee du foncteur. Il est preferable de parcourir les elements du vecteur par un iterateur. Passer des const Eleve& au foncteur. Ne pas lister explicitement toutes les lettres, utiliser leur consecutivite.

POUCIN	Florentin	4	Il vaut mieux cacher ptr dans la partie privee. Acceder au compteur par ce pointeur dans le main est une contorsion, alors que la variable est directement disponible. Il y avait mieux a faire qu'un tableau des lettres possibles. La comparaison des string existe deja operator<, et en plus ta reimplementation est incomplete : deux noms dont l'un est prefixe de l'autre provoquent un retour d'une valeur arbitraire.
QUILY	Ludovic	3.5	Faire int*compteur; *compteur=0; est un bug grave: on ecrit a une adresse arbitraire, celle pointee par compteur. Il est preferable de parcourir le vecteur par un iterateur. Passer des const Eleve& au foncteur.
RIOU	Auriane	4	Il vaut mieux parcourir le vecteur par un iterateur. Cacher le ptr dans la partie privee. Mettre const les methodes de Eleve et passer des const Eleve& a operator().
RUGET	Simon	3	Le programme plante, car tu ne geres pas les copies de foncteur, dont l'implementation par defaut ne convient pas a cause de l'allocation dynamique. Il faut s'y prendre autrement: une variable exterieure sur laquelle le foncteur se contente de pointer, sans faire d'allocation.
SANCHEZ CRUZ	Jenny	2.5	Il y a un gros probleme memoire: ton constructeur du foncteur fait pointer sur le parametre, qui est une variable locale car non passee par reference. Le get_compteur est inutile. Il vaut mieux utiliser un iterateur pour parcourir le vecteur.
SCHLEGEL	Nicolas	4	La fonction affiche_eleves devrait prendre un const vector<Eleve>& pour eviter une copie. Cacher ptr dans la partie privee du foncteur. Passer des const Eleve& au foncteur.
SOUSSI	Nada	4	Il y a de bonnes choses, mais une erreur memoire: le delete ptr dans le destructeur du foncteur est interdit car le constructeur fait pointer sur une variable exterieure non obtenue par new. C'est bien d'avoir passe des const Eleve& au foncteur et d'avoir bien implemente operator<< du vector.
STÄMPFLI	Erwan	4	Il faut borner le nombre max d'eleves. Passer des const Eleve& aux foncteurs. Ne pas lister explicitement toutes les lettres. D'ou vient ce bits/stdc++.h ? Ce n'est pas un fichier standard du C++, c'est lie au compilateur.

TRINH	Robin	1	Ca ne compile tout simplement pas car CompareNom n'a pas de constructeur sans argument (utilise dans AfficheNom). De plus, le constructeur du foncteur fait pointer sur une variable locale (son parametre), ce qui est une grave erreur. J'aimerais bien que les fichiers portent des noms plus appopries que Tennis et matrices...
VAN DEN BERGH	Candice	5	Il y avait mieux a faire que lister toutes les lettres, mais c'est tres bien par ailleurs.
VENARD	Paul-Louis	4	Faire la liste de toutes les lettres n'est pas optimal, il y avait bien plus simple. Ne pas implementer le constructeur par copie quand il ne fait rien de mieux que celui qui aurait ete genere. La fonction print devrait prendre un const vector&. Le tri s'effectue bien, contrairement a ce que dit ton commentaire. Ne pas faire un projet Qt.
VINCENT	Théo	4.5	Bon ensemble. Proteger le header contre la double inclusion. Ne pas lister explicitement la liste des caracteres.
VONGPASEUT	Clarine	4.5	Il etait possible de faire plus simple que de lister tous les caracteres. Bien sinon, mais passer des const string& est preferable.

Heritage(/5)

ALLAIN	Clément	4	Le champ _integral ne doit pas etre au niveau de function mais d'une sous-classe Derivee. Il faut une fonction clone pour que les fonctions puissent exister independamment, sans risque avec les pointeurs.
BAKONG EPOUNÉ	Killian	3	Il faut des destructeurs virtual pour les classes de bases. Un appel a derivee faisant un new, il faut le delete associe. Le constructeur par copie de Polynome et celui de Derivee ont des problemes car ils ne gerent pas correctement les pointeurs.
BEL KHAYAT ZOUGGARI	Yassine	2	Le programme plante a cause de pointeurs. Je ne suis pas sur que tu aies compris le principe de Derivee, ni l'interet de clone().
BERTRAND	Romain	4	Il manque un constructeur par copie a Polynome, necessaire car appele par clone. Son destructeur doit etre virtuel. Derivee::clone doit faire integrale->clone pour eviter une dependance sur le meme objet. Mettre les champs prives. Attention a faire un delete apres un appel a derivee() qui fait un new.
BIRAC	Nicolas	4.5	Les champs x_0 et x_1 sont superflus dans Affine, qui herite de Polynome et peut donc stocker dans son champ coefs. Le clone de Derivee est inutilement complique. Tout appel a derivee fait un new, donc il doit y avoir le delete correspondant.
BOUGHDIRI	Ahmed	3.5	Affine::derivee retourne l'adresse d'une variable locale, c'est un bug grave. Un appel a derivee ou a clone fait un new, il faut donc le delete correspondant. Les destructeurs de Function et Polynome doivent etre virtuels.
BRAMI	Victoria	1	Le code ne compile pas pour de multiples raisons: les methodes clone sont parfois definies const, mais pas toujours. Ca fait que certaines classes restent abstraites car elles n'implementent pas le clone de Function. Confusions entre objet pointe et pointeur a plusieurs reprises. Fonction::inverse, qui est au coeur de l'exercice, n'est pas implementee.
BRISINGER	Maxime	3.5	Les appels a derivee dans inverse font des new, il doit donc y avoir les delete correspondant. La gestion du tableau de Polynome est insuffisante, attention aux copies. Les methodes de Derivee sont bien confuses, avec les clones et les copies.

CANCES	Adrien	4	La function inverse appelle derivee qui fait des new. Il faut donc les delete correspondant. Le destructeur de Polynome doit etre virtuel car on en herite. Attention de bien appeler delete[] pour un tableau.
CHAKIR	Adel	1	Ou est le main? Le progamme ne compile pas car certaines methodes virtuelles pures de Fonction ne sont pas correctement redefinies par Derivee. Inutile de stocker les coefficients de Affine, ils peuvent se mettre dans les donnees heritees de Polynome.
CHAMBON	Loick	4.5	Le destructeur de Polynome doit etre virtuel. Attention au risque de confusion entre Derivee(const fonction&) et le constructeur par copie. Il vaut mieux avoir Derivee(const fonction*) pour bien savoir lequel est appele.
CHANCEREL	Pia	1	Ca ne compile pas a cause de multiples confusions entre pointeur et objet pointe. Dans inverse, l'appel a derivee fait un new, il faut donc le delete correspondant. Attention a la gestion memoire du tableau dans Polynome.
COSSON	Margot	4	Le constructeur par copie de Polynome doit etre defini pour gerer correctement la memoire. Attention a appeler delete apres derivee (qui fait un new) dans inverse. La definition de Polynome::inverse est inutile, puisqu'elle ne fait rien de plus que Fonction::inverse. Il y a un double clone dans Derivee::derivee. Attention, le constructeur par copie de Derivee ne clone pas integral. De plus, le destructeur de Derivee doit faire delete integral.
DESHORS	Charles	3.5	Le constructeur par copie de Polynome est necessaire pour gerer correctement le tableau des coefficients (appele par clone). Le constructeur par copie de Derivee ne fait pas un clone de integrale, donc risque de double delete. La methode inverse faisant appel a derivee et donc a new, il doit y avoir un delete.
DESJARDINS	Antoine	2	Il manque Trigo::derivee, qui est quand meme un point important de l'exercice. Fonction::inverse appelle derivee qui fait un new, il manque le delete.
DURIF	Anne	1.5	Le programme ne compile pas. Il faut bien respecter les const pour redefinir une methode d'une classe de base, sinon la classe reste abstraite. La fonction inverse doit faire delete suite a derivee qui fait un new.

FATOUX	Lambert	4	Il y a un double clone dnas Trigo::derivee car le parametre passe au constructeur de Derivee est lui-meme clone. Derivee::clone appelle son constructeur par copie (non defini) qui ne fait pas le clone de integrale et est donc dangereux. Fonction::inverse doit appeler delete sur la fonction retournee par derivee, acquise par new.
FAUDUET	Alex	5	TB. Juste deux petites fuites memoire des tableaux passes au constructeur de Polynome.
FERNANDEZ	Raphaël	2	Les methodes get_deg et get_coeff n'ont rien a faire dans Fonction, seulement dans Polynome. La fonction inverse ne doit pas forcement utiliser Derivee, en particulier pour un Polynome ca doit etre un nouveau Polynome. Il s'agit donc d'appeler la methode derivee, qui fera la difference. Il ne faut pas redefinir Polynome::inverse. La derivee seconde fait un crash du programme.
FRANCOIS	Quentin	4	Le constructeur par copie est necessaire pour Polynome pour gerer correctement le tableau coef. Derivee::derivee est deficient car il appelle un constructeur par copie non defini, et qui ne fait donc pas de clone. Il est necessaire pour inverse de faire le delete apres derivee, puisqu'un new a lieu.
GAINON	Louise	4	Le constructeur par copie de Polynome manque : il est appele par clone et ne gere pas correctement la memoire. Le destructeur de Derivee doit faire delete integrale. Le constructeur par copie de Derivee doit cloner le champ integrale pour ne pas avoir double pointage. Fonction::inverse doit faire un delete apres derivee qui fait un new.
GINOULHAC	Raphaël	5	TB, juste une fuite memoire dans Polynome::derivee : le tableau servant a construire les coefficients n'est pas libere.
HÉMADOU	Louis	4	Mauvaise gestion de la memoire de Polynome car le constructeur par copie appele par clone n'est pas defini. Faire un delete apres derivee dans inverse, car cela fait un new.
LAINÉE	Martin	1	Ca ne peut pas compiler, car Polynome reste une classe abstraite puisqu'elle ne definit pas la methode virtuelle pure Fonction::clone.

LANDAIS	Jean	3.5	Le destructeur de Polynome doit etre virtuel car on en herite. La classe Derivee doit utiliser clone pour ne pas dependre d'un champ integrale qu'elle ne controle pas. Il ne faut pas redefinir Polynome::inverse, celui de Fonction fait exactement ce qu'il faut (sauf qu'il oublie d'appeler delete sur le resultat de derivee...)
LASRY	Raphaël	3.5	Derivee::derivee fait un double clone. Polynome::inverse n'a aucune raison d'etre defini. L'appel a derivee dans inverse doit etre suivi d'un delete car derivee fait un new. Le destructeur de Derivee doit faire delete integrale.
LÊ	Paul-Vinh	3.5	L'interet de clone n'est pas compris, il prend tout son sens dans Derivee. Affine ne doit pas definir de nouveaux champs mais profiter de ceux herites de Polynome. Fonction::inverse appelle derivee qui fait un new, donc il manque le delete.
LE GRELLE	Hugues	3	Il y a une confusion avec derivee. Ainsi celle de Trigo retourne juste un clone, qui est ensuite mis dans Derivee B=A.derivee(). Cette instruction appelle le constructeur de Derivee prenant une Fonction*, ce qui produit le bon resultat, mais n'est pas le mecanisme logique. La gestion memoire de Polynome est deficiente car le constructeur par copie n'est pas defini.
LESUEUR	Louis	4	Le destructeur de Polynome devrait etre virtuel. Derivee::clone appelant le constructeur par copie non defini ne fait pas ce qu'il faut, c'est-a-dire un clone de integrale. Le constructeur par copie de Polynome manque. La fonction inverse doit appeler delete suite a derivee qui fait un new.
LOISON	Virginie	2.5	Pourquoi le programme n'affiche-t-il quasiment rien ? Il manque un constructeur par copie pour Polynome, necessaire pour bien gerer le tableau. Le destructeur de Polynome doit faire un delete. Affine ne devrait pas ajouter des champs pour stocker les coefficients. Inverse n'a pas besoin de faire un clone pour prendre les valeurs. Le fonctionnement de Derivee est tres confus.
MEDINA	François	1.5	Le main est vide. Rien n'a ete teste ? Dans la fonction inverse, c'est bien d'avoir pense a delete f_prime, mais ca vient apres le return, donc jamais appele ! Il manque un construceur par copie pour Polynome afin de gerer correctement le tableau.

MOREL	Sébastien	3	Il y a des problemes dans la gestion memoire de Polynome, le constructeur par copie, utilise par clone, n'etant pas defini. Pour inverse, inutle de cloner la fonction a evaluer. Par contre, il ne faut pas oublier de faire un delete car derivee fait un new. Derivee::clone appellle le constructeur par copie, non defini, qui ne fait pas un clone du champ integrale. Inutile de rajouter des champs pour a et b de Affine, qui herite de Polynome.
OUHAÏCHI	Firas	4	Derivee::clone appelle le constructeur par copie, non defini, et qui ne fait donc pas un clone de integrale. Il manque des destructeurs virtuels pour Fonction et Polynome. Fuite memoire dans inverse, car clone et derivee font un new.
PAN	Chao	4	La methode inverse ne doit pas faire un difference finie, mais appeller derivee(), qui fera un calcul plus precis dans le cas de Polynome. Derivee:clone appelle le constructeur par copie, ce qui n'est pas bon car il n'est pas defini. Il devrait cloner le champ integrale.
PION	Aurélien	4	La fonction inverse oublie de faire le delete, necessaire car derivee fait un new. Le destructeur de Derivee devrait faire delete integrale. Le clone de Derivee appelle son constructeur par copie, non defini et qui ne fait pas de clone de integrale, d'ou double pointeur sur la meme zone memoire.
POLETTE	Nadège	3.5	Fonction n'a pas de raison d'avoir un champ integrale, dont on herite par exemple dans Polynome et qui ne sert a rien pour cette classe. Dans la classe Derivee, les methodes derivee et Clone appelllent Derivee(*this), constructeur par copie non defini, au lieu de Derivee(this), qui ferait un clone comme il faudrait. La fonction inverse appelant derivee et donc new doit faire un delete.
POLI	Maxime	4.5	Il est delicat d'avoir les constructeurs de Derivee prenant un Fonction* et l'autre un Derivee*, le premier faisant une derivation et le deuxieme se comportant comme un constructeur par copie. Le polymorphisme ne jouant pas de facon descendante sur les arguments, on peut s'attendre a des surprises. Derivee::derivee fait un double clone, d'ou fuite memoire.

POUCIN	Florentin	3	Il manque un destructeur virtuel pour la classe fonction et celui de Polynome devrait etre virtuel. La fonction inverse n'a pas besoin de cloner, par contre elle doit desallouer la memoire allouee par derivee(). Polynome::clone appelle son constructeur par copie non defini, qui ne gere pas bien son tableau. Derivee::derivee fait un double clone, d'ou fuite memoire.
QUILY	Ludovic	4.5	Il manque simplement la desallocation suite a l'appel a derivee() dans inverse. Avec cette correction et les delete dans le main, on obtient un programme sans erreur et sans fuite memoire.
RIOU	Auriane	2	Le programme plante a cause du constructeur par copie de Polynome qui n'est pas defini et ne gere donc pas correctement le tableau interne. Pas mal de confusion avec des clone de clone.
RUGET	Simon	4	inverse doit faire un delete dF car celui-ci est obtenu avec un new. Polynome::clone a une fuite memoire avec le tableau coef. Les champs a et b de Affine sont superflus, on herite du tableau coefficient de Polynome. Trigo::derivee fait un drole de mic mac, c'est bcp plus simple que ca.
SANCHEZ CRUZ	Jenny		
SCHLEGEL	Nicolas	3.5	Il manque un delete dans Fonction::inverse, qui appelle derivee et fait donc un new. Pas de tableau statique de taille variable dans Polynome::derivee, c'est une extension du C++ qui n'est pas standard et dangereuse. Le destructeur de Derivee devrait faire delete integrale.
SOUSSI	Nada	1.5	Il manque la classe Derivee, qui est au coeur de l'exercice. Trigo::derivee ne retourne pas ce qu'il faut.
STÄMPFLI	Erwan	4	Il manque le constructeur par copie a Polynome, appele par clone et non defini, et ne gerant pas correctement le tableau interne. Fuites memoire dans Polynome::derivee, il faut desallouer les tableaux temporaires.
TRINH	Robin	2	L'idee est de factoriser la methode inverse le plus haut possible, dans Fonction. La classe Derivee est incomplete. Le principe du clone n'est pas compris. Nommer les fichiers Tennis.cpp Matrices.h et Matrices.cpp pour un TP qui n'a rien a voir releve de la paresse.

VAN DEN BERGH	Candice	4.5	Dans Fonction::inverse, il ne faut pas construire un objet Derivee, qui fait des differences finies, mais utiliser la methode derivee(), qui pour un polynome donnera une fonction plus precise. Utilisation de clone un peu maladroite a plusieurs reprises. Utiliser delete[] pour un tableau. Sinon, pas de fuite memoire, bravo !
VENARD	Paul-Louis	2.5	Polynome::derivee est fausse a plus d'un titre : la boucle deborde du tableau coeff, et la methode renvoie un pointeur sur une variable locale ! Tout appel a derivee doit faire un delete pour eviter une fuite memoire.
VINCENT	Théo	2.5	Les accesseurs au niveau de Fonction n'ont aucun sens. Je ne vois pas l'utilite du constructeur de Polynome prenant Function* en argument. Et si l'argument n'est pas un Polynome ? Les appels a derivee() doivent etre suivis d'un delete pour eviter la fuite memoire.
VONGPASEUT	Clarine	4	Les appels a derivee doivent etre suivis d'un delete, car il y a allocation memoire. Ecrire new Polynome(Polynome(...)) est tres maladroit, cela cree un Polynome anonyme qui est utilise comme argument du constructeur par copie. Derivee::derivee() fait un double clone.

Data

		Data(/5)	
ALLAIN	Clément	5.5	Bravo, y compris pour la doc ! Conseil: quand tu definis une macro qui a plusieurs instructions (comme ton TEST), il vaut mieux entourer de {} pour que ca se comporte comme une seule instruction. Sinon, avec du code comme while(...) TEST(...); le try/catch a lieu une seule fois apres la boucle des cout, ce qui devrait surprendre le programmeur.
BAKONG EPOUNÉ	Killian	2.5	Le test des exceptions n'est pas fait. Ne jamais appeler un destruteur explicitement.
BEL KHAYAT ZOUGGARI	Yassine	4	Il manque le parcours en largeur. Le rapport est un peu succint.
BERTRAND	Romain	4	Correct dans l'ensemble. Le fait qu'on ne puisse pas separer le template en 2 fichiers n'a rien a voir avec de l'allocation memoire.
BIRAC	Nicolas	4.5	Bon ensemble, mais try/catch ne devraient pas etre dans insertSon et removeSon, elles devraient laisser l'exception remonter a l'appelant, qui saura peut-etre quoi en faire.
BOUGHDIRI	Ahmed	5.5	TB. Bravo pour la doc doxygen.
BRAMI	Victoria	4	removeLastSon et removeSon doivent faire un delete. Faire delete &data est une erreur potentiellement grave, car data n'a pas ete obtenu par new. Pas de [] dans le delete pour un pointeur non tableau. Ta class Noeud aurait pu etre simplement un pair<int,int>.
BRISINGER	Maxime	3.5	snprintf doit prendre un tableau de caracteres deja alloue poue ecrire dedans. De plus, snprintf avec %d fait que seul Tree<int> pourrait afficher correctement. Le destructeur de Tree n'est pas defini. Les remove doivent desalllouer.
CANCES	Adrien	4.5	Les remove doivent faire un delete. Le destructeur doit faire des delete pour detruire le sous-arbre, on n'appelle jamais explicitement un destructeur. Il aurait ete bien de provoquer les exceptions dans le main pour verifier. Tes minDepth et maxDepth font le parcours en profondeur, non en largeur. C'est bien pour maxDepth, on peut faire mieux pour minDepth.
CHAKIR	Adel	3.5	Les remove ont besoin de faire un delete. Le destructeur n'est pas defini.
CHAMBON	Loick	4	Code plutot bien, mais il manque les delete des fonctions remove. Quand a la gestion des erreurs, il y a plusieurs points faux dans le rapport.

CHANCEREL	Pia	4.5	Dommage que tu n’aies pas implemente maxDepth, c’etait plus facile que minDepth, qui est presque bon, a une erreur pres d’indice dans le tableau ordre. Attention, ne jamais appeler un destructeur, le delete s’en charge deja.
COSSON	Margot	4.5	Les methodes remove doivent faire un delete. Une autre methode de signalement d’erreur (a eviter) est errno. Bon ensemble.
DESHORS	Charles	4	On n’appelle jamais un constructeur explicitement, le delete s’en charge (entre autre). Les methodes remove doivent aussi faire un delete. Il est dommage que le main ne provoque pas volontairement des exceptions pour verifier leur interception.
DESJARDINS	Antoine	4	Le main devrait tester la levee d’exception et leur interception. La methode removeLastSon doit faire un delete.
DURIF	Anne	4	Retourner des exceptions sous forme de valeur numerique n’est pas tres judicieux. Il aurait aussi fallu provoquer ces exceptions dans le main. Les remove doivent faire un delete. Tu sembles confondre parcours en largeur et en profondeur. Dans le parcours en largeur, tu n’as pas besoin de 2 files, une suffit.
FATOUX	Lambert	4	La logique du destructeur m’echappe. MinDepth semble plus complexe que necessaire, mais l’idee est comprise. Les methodes remove doivent faire un delete. A supposer que vector::pop_back leve une exception, ce n’est pas a Tree de la gerer mais a la fonction appelante. Pas de catch donc dans Tree.
FAUDUET	Alex	5	Il manque simplement delete dans les methodes remove. Bien pour maxDepth et minDepth.
FERNANDEZ	Raphaël	4	Les remove doivent faire un delete. Le destructeur est bien plus complexe que necessaire. Le rapport avec Pages n’est pas portable.
FRANCOIS	Quentin	4	remove doit faire un delete. Le principe de mindepth n’est pas compris.
GAINON	Louise	3.5	Pas de rapport ? Remove doit faire un delete. Tree<T>::data est de type T, non int !Bien pour la doc.
GINOULHAC	Raphaël	4.5	On peut faire minDepth avec une seule file. Les methodes remove doivent faire un delete.
HÉMADOU	Louis	3	Les remove doivent appeler delete. Le destructeur ne doit pas faire delete this, ca n’a pas de sens. Par contre il doit faire un delete des sons et il ne faut pas appeler explicitement le destructeur.

Data

LAINÉE	Martin	0	Tu as mis le code du TP3
LANDAIS	Jean	5	Bon rapport et code avec tres peu de defauts: l'appel recursif a display doit passer a nouveau indent comme argument, et les commentaires doxygen doivent etre devant l'en-tete de la fonction et non juste apres (contrairement a python).
LASRY	Raphaël	4	setSon et removeLastSon doivent faire un delete pour eviter une fuite memoire. Il aurait ete bien de me fournir un main ou au moins une partie du code est active.
LÊ	Paul-Vinh	5	removeLastSon doit faire un delete. MinDepth est plus efficace avec un parcours en largeur, car on peut s'arreter des qu'une feuille est atteinte, on peut ne pas explorer l'arbre entier.
LE GRELLE	Hugues	4	Le main ne teste pas la classe template. Les methodes setSon et removeSon doivent faire un delete. Il est preferable de retourner une sous-classe de exception plutot qu'un string.
LESUEUR	Louis	2.5	Le code a-t-il ete teste ? En tout cas, il ne compile pas a cause de 2 erreurs, un point-virgule manquant et vector::erase avec un eniter en argument au lieu d'un iterateur. Comportement curieux de removeSon avec son appel recursif. En tout cas, il lui manque un delete, tout comme a setSon.
LOISON	Virginie	2	Le code ne compile pas a cause de problemes dans la partie template. Un destructeur ne s'appelle jamais explicitement. SetSon et removeLastSon doivent faire un delete. Pourquoi ajouter noSons() comme methode alors que nbSons() fait la meme chose ?
MEDINA	François	3.5	Pas de rapport. SetSon et removeLastSon doivent faire un delete.
MOREL	Sébastien	4.5	setSon et removeLastSon doivent faire un delete. Une erreur dans minDepth: il ne faut pas forcement augmenter depth des qu'on n'est pas feuille, le suivant peut etre un frere! Il se trouve que ton arbre de test ne declenche pas ce bug. Il aurait ete bien de tester les exceptions. Bon rapport par ailleurs.
OUHAÏCHI	Firas	2.5	Le main est minimal et ne teste pas les exceptions, entre autre. Ne jamais appeler explicitement un destructeur. Tree<T> n'est pas teste.
PAN	Chao	5	Bon travail. Il manque juste un delete dans setSon et dans removeLastSon.

PION	Aurélien	4	Le rapport semble non termine. Il y a un probleme de link pour IntTree::displayNew. Ce n'est pas aux mehtodes de Tree de gerer les exceptions, elles doivent juste en lever quand il le faut. Il manque des delete pour setSon et removeLastSon. Il aurait ete bien que le main teste les exceptions une par une avec des catch.
POLETTE	Nadège	5.5	Excellent ! Il y a juste le return 0 a la fin de minDepth qui est un peu deroutant, mais qui n'est en fait jamais atteint. Pour le signaler au programmeur, il est bien de faire par exemple assert(false) juste avant.
POLI	Maxime	5.5	C'est tres bien ! Encore mieux aurait ete de provoquer chaque exception tour a tour dans le main et recuperer par catch. A noter que la queue<int> dans minDepth gache de la memoire, on pourrait s'en passer. Quitte a ne pas s'en soucier, il est plus elegant de faire une seule queue<pair<Tree<T>*,int>>.
POUCIN	Florentin	3.5	Il manque des delete dans setSon, removeLastSon, etc. Ne jamais appeler le destructeur explicitement. Il est dommage que la classe template ne soit pas instanciee et testee, un bug de display aurait pu etre detecte....
QUILY	Ludovic	4.5	Il manque des delete dans setSon et removeLastSon. sons[i] n'est pas un tableau, donc il faut simplement un delete sans []. Il est plus efficace pour min_depth de faire une traversee en largeur pour stopper des qu'on rencontre une feuille.
RIOU	Auriane	4.5	setSon, removelastSon et removeSon devrait faire un delete.
RUGET	Simon	5	Il manque des delete dans setSon, removeSon et removeLastSon. La queue<int> dans minDepth est un peu superflue, car on n'a pas plus de deux profondeurs presentes dans la file a tout moment.
SANCHEZ CRUZ	Jenny		
SCHLEGEL	Nicolas	4.5	setSon et removeSon devraient faire un delete. Ca ne sert a rien de lever une exception pour l'intercepter dans la meme fonction, une variable suffirait pour ca. Renvoyer des string est moins bien qu'une exception, car on ne peut pas trier par type.
SOUSSI	Nada	5	Il manque un delete dans removeSon. Le queue<int> dans minDepth aurait pu etre omis en utisant juste 2 variables.
STÄMPFLI	Erwan	2	Il manque la gestion des erreurs, le test de arbregen.

TRINH	Robin	4	Ne pas appeler de destructeur explicitement. SetSon et removeSon devraient faire un delete. Les exceptions ne sont pas testees.
VAN DEN BERGH	Candice	4.5	getSon devrait aussi lever une exception. Ce n'est pas a removeSon d'attraper l'exception qu'elle a elle-meme levee, c'est a l'utilisateur de recuperer l'erreur. Display oublie de passer a nouveau indent dans l'appel recursif.
VENARD	Paul-Louis	3.5	setSon, removeSon et removeLastSon devraient faire un delete. Lever des exceptions est bien mieux que cerr suivi de exit, car on peut recuperer l'erreur et prendre des mesures adequates.
VINCENT	Théo	3.5	cf Maxime Brisinger
VONGPASEUT	Clarine	4	Il manque delete dans getSon, removeSon et removeLastSon. Il est bizarre que setSon lance une exception de type string ou out_of_range, ca manque d'uniformite, Il est maladroit de passer profondeur dans display, un appel recursif avec prefix modifie est plus simple.

		ProgDyn(/5)	
ALLAIN	Clément	4.5	Bon travail dans l'ensemble. Tu as 2 bugs memoire dans DISTANCE4: next=cur+1%ROW_CNT il manque les parentheses (cur+1) car % est prioritaire (comme une division); Pour cur-1%ROW_CNT, il faut non seulement les parentheses, mais aussi s'assurer de ne pas etre negatif (modulo d'un negatif est negatif en C++), donc il faut ecrire (cur+ROW_CNT-1)%ROW_CNT.
BAKONG EPOUNÉ	Killian	4.5	Ne pas mettre la definition des fonctions dans les .h. Il manque l'affichage de la liste des operations. Bien par ailleurs.
BEL KHAYAT ZOUGGARI	Yassine	3	Il manque l'analyse theorique de la complexite et la variante Damerau, qui etait pourtant facile a ajouter.
BERTRAND	Romain	4	L'algorithme recursif n'a pas une complexite exponentielle en espace, car les branches explorees liberent leur memoire. En fait, c'est juste la profondeur de la recusion qui determine la complexite en espace, et elle est lineaire en la longueur des mots. Le code d'affichage du chemin est fastidieux.
BIRAC	Nicolas	4.5	La complexite nulle en espace de l'algo recursif est illusoire, car il y a la profondeur de la recursion qui compte, donc lineraire. Le code est plutot bien, mais il aurait ete souhaitable de bien separer l'affichage du calcul du tableau.
BOUGHDIRI	Ahmed	4	Il y a un probleme memoire dans l'affichage quand le mot 1 est un prefixe du mot 2. Il est dommage d'avoir reecrit tout pour la variante Damerau, il suffisait d'une variable pour eviter de se repeter.
BRAMI	Victoria	4	Il manque l'affichage des transformations (le code commente plante). La complexite spatiale de l'implementation recursive n'est pas nulle, puisqu'il y a la profondeur de la recursion qui joue.
BRISINGER	Maxime	4	Ce n'est pas une tres bonne idee d'avoir une variable globale (le tableau de memoization). Damerau est juste une variante, qui aurait pu se gerer par une simple variable dans les fonctions.
CANCES	Adrien	4.5	Dans le code, on peut s'affranchir du cout d'extraction de sous-chaines en passant juste un indice indiquant le nombre de lettres a prendre en compte. Damerau etant juste une variante, le code aurait pu gerer ca par une variable. Sinon, tres bien.

CHAKIR	Adel	3.5	Le rapport est minimal et ne repond pas a toutes les questions. Le code est globalement correct, mais il comporte des fuites memoire, avec une allocation juste avant la desallocation pour une zone memoire deja allouee avant. Prendre des prefixes de facon repetee a un cout, dont on peut se passer avec des indices.
CHAMBON	Loick	4	Retourner les mots pour un affichage dans l'ordre est une possibilite, mais il y a mieux avec une fonction recursive qui affiche "en remontant". Il est faux que la complexite en espace de la methode recursive sans memo est exponentielle, elle est lineaire.
CHANCEREL	Pia	4	Le code est bien. L'affichage des etapes peut se faire bcp plus simplement par une fonction recursive. Tu as bien repondu aux questions 5.2 et 5.3 dans le rapport, leur implementation n'etait pas si compliquee, c'est dommage.
COSSON	Margot	4	Damerau etant juste une variante, il est dommage d'avoir quasi duplique le code plutot que d'utiliser une variable. L'affichage des etapes peut se faire avec bcp moins de code avec une fct recursive.
DESHORS	Charles	3.5	Il manque la preuve de la sous-structure optimale. Pour extraire un prefix d'une string, il y a une methode pour ca. La variante Damerau etant presque identique, il vaut ;ieux gerer par une variable. Le main est trop minimal.
DESJARDINS	Antoine	4	Il faut toujours bien distinguer complexite en temps et en espace. La version recursive a une complexite lineaire en espace. Le code est bien, il manque juste l'affichage des etapes.
DURIF	Anne	3.5	Il y a clairement des problemes dans le code, car il affiche une distance a -1 et d'autres a 0. Dans levenam, c'est du a F[n1*n+n2] au lieu de F[n1*(n+1)+n2]. Il manque l'affichage des etapes.
FATOUX	Lambert	4	Ne pas definir les fonctions dans un .h, juste les declarer. Il y a plus direct que d'enregistrer le chemin optimal pour affichage. La complexite spatiale de la version recursive n'est pas exponentielle, car les branches deja explorees ne prennent plus de memoire.
FAUDUET	Alex	4	La complexite spatiale sans memo n'est pas vraiment O(1) puisqu'il y a les variables de toute la pile d'appel. Elle est donc proportionnelle a la profondeur de recursion, soit linearie. Pour l'affichage, on peut faire meiux avec une fonction recursive.

FERNANDEZ	Raphaël	3.5	Il est normalement interdit d'avoir un tableau statique de taille variable. La complexite de l'algo recursif avec memo n'est pas exponentielle en temps (ni en espace) car on ne recalcule pas deux fois la meme distance.
FRANCOIS	Quentin	4	Il faut bien faire la distinction entre complexite en temps et espace dans le rapport. Pour le calcul des etapes, il aurait ete preferable de faire une methode recursive sans stockage explicite du chemin.
GAINON	Louise	4.5	Bon rapport. A noter que la complexite spatiale sans memoization est quand meme lineraire, puisqu'il y a la profondeur de la recursion qui intervient. Pour l'affichage des transformations, c'est exactement ce qu'il faut faire dans le principe, mais il y a quelques bugs dans l'implementation. En particulier, il faut afficher "en remontant".
GINOULHAC	Raphaël	4.5	La version recursive sans memo n'a pas une complexite exponentielle en espace, les branches explorees n'occupent plus de memoire. Bien pour le code. Pour l'affichage des transformations, plutot que d'avoir une pile explicite, on peut la faire recursive.
HÉMADOU	Louis	4	Il y a une erreur sur la complexite en espace sans memo, elle est lineaire, non exponentielle car les branches explorees n'occupent pas de memoire. Il manque l'affichage des etapes.
LAINÉE	Martin	3.5	Il manque la variante Damerau, l'affichage des etapes. Le main devrait tester tout ce qui a ete implemente.
LANDAIS	Jean	3.5	Le rapport est soigne mais comporte une erreur sur la complexite en espace de l'algo naif. Cette complexite est en fait lineaire, car les branches explorees n'occupent plus de memoire (les appels recursifs se font sequentiellement, pas en paralllele). Dommage qu'il manque Damerau, c'etait facile a ajouter. Ne pas utiliser de variable globale pour la memo. Rien ne s'affiche concernant les etapes.
LASRY	Raphaël	4	Bien distinguer dans le rapport complexites en temps et espace, elles sont tres differentes dans le cas recursif sans memo. Ta classe n'ayant pas de donnees, que des methodes, elle aurait plutot du etre un namespace. On peut faire mieux dans l'affichage, mais l'idee de le faire recursif est la bonne !

LÊ	Paul-Vinh	4	Le rapport est soigne. La complexite en espace n'est pas exponentielle, car les branches explorees n'occupent plus de memoire. Il est dommage que Damerau n'ait pas ete integre, il suffisait de rajouter une variable aux fonctions. Pour l'affichage, le probleme est qu'on part de la fin, il faut donc "retourner" les etapes. Une fonction recursive fait ca tres bien. Attnetion de bien faire les delete apres des new.
LE GRELLE	Hugues	3.5	Ta fonction recurcivemem n'ecrit pas dans M, donc en fait elle ne fait pas mieux que la version directe. L'implementation de Damerau devrait etre le meme code avec juste une variable bool. L'affichage peut se faire de facon bcp plus propre. Le rapport devrait justifier les complexites annoncees.
LESUEUR	Louis	3.5	C'est bien pour les timings, mais les complexites peuvent aussi tres bien s'analyser de facon theorique. La complexite spatiale de la version recursive simple est en fait lineaire, non exponentielle. Il est dommage de ne pas avoir garde un code commun pour la version Damerau, avec juste une variable. Pour l'affichage des operations, on n'est pas oblige de stocker pendant la recherche.
LOISON	Virginie	4	Il ne faut pas utiliser de variable globale, meme pour la memoization. Il est important d'aerer le code, il est tres indigeste tel quel. Pour l'affichage du chemin, il y a mieux a faire, le tableau de memo suffit entierement.
MEDINA	François	2.5	Appeler les fonctions auxiliaires alpha et beta n'est pas tres informatif. Il y a bien mieux a faire pour l'affichage du chemin. Le rapport est minimal, tout comme le main.
MOREL	Sébastien	3.5	Il manque la complexite en espace dans la methode sans memoization. Il y a beaucoup de repetitions dans le code, qui aurait pu etre mieux organise. Il manque la version iterative avec complexite memoire minimale. Les tableaux alloues dynamiquement ne sont jamais liberes, fuite memoire.
OUHAÏCHI	Firas	3	Le rapport est minimal et ne justifie pas en particulier la sous-structure optimale. Il manque l'affichage des changements et la version recursive avec complexite en espace optimisee. Les variantes Damerau auraient du se gerer par une simple variable, plutot qu'une repetition de code quasi identique.

PAN	Chao	3.5	La version avec memoization n'est pas plus rapide que celle sans, car tu n'utilises pas les valeurs deja calculees pour eviter des appels recursifs inutiles. La complexite en espace de la version sans memoization n'est pas constante mais lineaire, car il y a la profondeur de la recursion.
PION	Aurélien	3.5	La complexite en espace est lineaire dans la version recursive sans memo (profondeur de la recursion). La preuve de sous-structure optimale n'est pas explicite. Le code laisse beaucoup de fuites memoire. Il manque la version iterative avec memoire minimale.
POLETTE	Nadège	4.5	Bon rapport. Le code est bien mais il y a un bug dans dist_lev_it_lin: le return line0[chain2.size()] peut ne pas exister s'il s'agit du line1 initial, ca depend du nombre de swaps effectues dans la boucle.
POLI	Maxime	5	TB. Il manque juste la complexite en espace de la methode directe, qui est lineaire.
POUCIN	Florentin	4	Le rapport est un peu succint. Bien distinguer complexites temporelle et spatiale. Le code aurait pu comparer des mots plus longs et plus eloignes pour avoir des temps de calcul plus significatifs. La variante Damerau aurait pu se faire avec une variable plutot que du code quasi duplique.
QUILY	Ludovic	4	Il faut bien distinguer complexites temporelle et spatiale. On n'a pas le droit d'avoir un tableau statique de taille variable normalement (meme si le compilateur l'autorise, c'est dangereux car peut echouer pour des tailles trop grandes). La variante Damerau aurait pu se gerer par une variable passee aux fonctions.
RIOU	Auriane	4.5	La sous-structure optimale n'est pas vraiment prouvee dans le rapport. La complexite en espace de la version sans memoization n'est pas nulle, il y a la profondeur des appels recursifs a prendre en compte. Le code est bien ecrit et clair.
RUGET	Simon	4.5	OK pour la partie theorique. Le code est bien, mais la partie sur l'affichage du chemin est tres fastidieuse et plus complexe que necessaire. Dans distance_2, il est normalement interdit d'avoir un tableau statique de taille variable.
SANCHEZ CRUZ	Jenny		
SCHLEGEL	Nicolas	5	TB. Le code aurait pu etre un peu plus compact pour DL (juste un terme a ajouter dans le min).
SOUSSI	Nada	3.5	Le code est bon, mais il manque le rapport.

STÄMPFLI	Erwan	4	Le rapport est bien, mais la complexite en espace de la version recursive n'est pas exponentielle mais lineaire, car les iterations n'ont pas lieu simultanement. Le code est correct bien que trop long, mais il ne faut pas utiliser de variable globale (F) et les tableaux statiques de longueur variable sont theoriquement interdits (bien que ton compilateur les autorise).
TRINH	Robin	4	Le rapport est correct. Dans le code, ne pas utiliser de variables globales (pour la memoisation). Les instructions suivant un return ne sont jamais atteintes, donc tes delete ne sont pas effectues, d'ou fuite memoire.
VAN DEN BERGH	Candice	4.5	Bon ensemble. Il aurait ete bien de separer la fonction d'affichage. Il n'est pas necessaire de stocker le chemin pour l'affichage, un appel recursif fait la meme chose plus simplement. Bien distinguer complexite en temps et espace dans le rapport.
VENARD	Paul-Louis	3	La complexite en espace n'est pas exponentielle, les appels ne sont pas simultanes. Le code recursif avec memoization doit utiliser les valeurs deja calculees, pas se contenter de juste enregistrer. Ton main a des tableaux statiques de taille variable, interdits par le standard.
VINCENT	Théo	3.5	La complexite en espace ne peut pas etre exponentielle, car les appels ne sont pas simultanes. Ajouter Damerau pouvait se faire facilement en ajoutant un terme dans le min. Ta tentative avec Operation aurait pu marcher, mais il aurait deja fallu un destructeur virtuel pour Operation (classe dont on derive). Stocker les antecedents rend le code un peu confus, alors qu'on peut facilement retrouver avec les distances de prefixes.
VONGPASEUT	Clarine	4	La complexite en espace de la version recursive n'est constante qu'en apparence. En realite, il y a le cout de la recursion, et donc elle est proportionnelle a la profondeur des appels, c'est-a-dire lineaire en la taille des mots. Il faut une fonction recursive avec affichage "en remontant" pour la suite des operations elementaires. Il manque la partie avec espace minimal et memoization.

QuadTree(/5)

ALLAIN	Clément	5.5	C’est tres bien. Ca manque un peu d’affichage (!) mais bon. A noter que tu as une fuite memoire dans gs_image::load (tableau data). D’autre part, les allocations de stb_image sont par defaut avec malloc/free (version C), ce qui n’est pas la meme chose que new/delete (version C++), il faut harmoniser.
BAKONG EPOUNÉ	Killian	4	Tu ne calcules pas le taux de compression. Il aurait fallu remplir l’image rectangulaire sans l’agrandir. Ce n’est pas une tres bonne idee de melanger feuilles globales en DAG et feuilles allouees a la demande, ca compliquerait la liberation memoire. La variable protect_leaves est globale, associee a un type.
BEL KHAYAT ZOUGGARI	Yassine	4	Le taux de compression ne depend pas que du nombre de feuilles. Les noeuds comptent aussi (et meme plus, car ils sont plus gros, et ils ne sont pas partages en DAG). Il aurait fallu aussi remplir les images rectangulaires sans les agrandir. Il faut faire des delete.
BERTRAND	Romain	3	Pour acceder a un pixel, x*width+y est absurde car width est la borne des x, c’est y*width+x. Ca explique l’affichage de l’image sombre. Le taux de compression est faux et provoque d’ailleurs une erreur memoire: int* taille=new int[0]; taille[0] n’existe pas puisqu’il y a 0 element dans le tableau. Tu n’as pas compris l’usage de protect_leaves et du DAG.
BIRAC	Nicolas	3.5	Le taux de compression est faux, il faut prendre en compte la taille des noeuds et des feuilles avec sizeof. Dans la reconstruction, le test i*height+j<height*width n’est pas suffisant, ca pourrait deborder en j quand i est petit. De plus, c’est j*width+i dont on a besoin puisque j est ton indice de ligne y. Tu as voulu bien faire avec les delete lorsqu’on recupere une feuille dans l’encodage, malheureusement les feuilles black et white sont partagees, donc detruites plusieurs fois. Ca fait planter le programme chez moi.
BOUGHDIRI	Ahmed	0	Plagiat: la plus-value par rapport au devoir d’Adrien Cances est minime, et ce n’est pas seulement la partie lecture/ecriture fichier qui est commune.

BRAMI	Victoria	4	Les conventions pour tes images rectangulaires sont deconcertantes, avec x l'indice de ligne et y l'indice de colonne, mais bon, ca reste coherent en interne. Tu ne calcules pas les taux de compression. Il manque les delete pour les pixels des images.
BRISINGER	Maxime	4.5	Le taux de compression n'est pas exact, car la taille d'un QuadTree n'est pas celle d'un QuadNode et QuadLeaf a encore une taille differente. Il faut donc distinguer, d'autant que les feuilles sont partagees dans certains cas. Melanger les feuilles partagees et allouees a la demande n'est pas une bonne idee si on voulait faire le menage ensuite.
CANCES	Adrien	5.5	Tres bon travail. Tu aurais pu aussi calculer le taux de compression en memoire avec des sizeof. En fait, tu as un peu de marge pour compresser plus: en appliquant simplement gzip a ton fichier d'arbre, je gagne un facteur 5 dans sa taille pour l'image binaire. Ce n'est pas de la triche, le format png fait la meme chose en interne avec la bibliotheque zlib. A noter quand meme un bug memoire avec display_squares. Egalement un bug dans get_tree_from_file: delete [] content suivi de aux_get_tree_from_file(content...)
CHAKIR	Adel	2.5	Il manque le calcul du taux de compression. L'interface de decode_image est etrange, avec une image en parametre eventuellement reallouee. encode_image_gris ne marche qu'avec des images carrees ? Le programme plante avec une image rectangulaire, meme binaire.
CHAMBON	Loick	4.5	Une erreur bete provoquant un bug memoire dans le test de l'image rectangulaire: le dernier parametre de Image_To_Qtree est I.W au lieu de I.H. Il y a des fuites memoire avec le contenu de l'image I ecrase avant que le tableau precedent soit detruit.
CHANCEREL	Pia	4.5	Pour le taux de compression, c'est plus complique que ca, il faut prendre en compte la taille des noeuds avec sizeof. Ce n'est pas une bonne idee de melanger feuilles partagees et feuilles allouees a la demande, ca compliquerait franchement le nettoyage memoire.
COSSON	Margot	4.5	C'est du bon travail, avec un rapport interessant. Le taux de compression est cependant inexact: les feuilles sont partagees, donc elles ne comptent pas. Par contre, ce sont les noeuds qui prennent la majeure partie de la memoire.

DESHORS	Charles	4	Ton CmakeLists est faux, il utilise le mauvais fichier. Le taux de compression n'est pas exact, il faut prendre en compte la taille des objets avec sizeof. Il faut qu'au niveau de l'extraction on n'agrandisse pas l'image.
DESJARDINS	Antoine	1.5	Le programme plante meme pour une image carree et binaire. L'acces au pixel se fait en y*width+x.
DURIF	Anne	3.5	Le taux de compression n'est pas correct. Ce qui compte, c'est surtout le nombre de noeuds, surtout quand on utilises le DAG. De plus, il faut prendre en compte qu'un noeud coute bien plus qu'un pixel, il faut utiliser sizeof pour savoir combien. Que vient faire la valeur 262000 dans le code, et les tailles 512 ? Il faut absolument utiliser width et height. Par ailleurs, le pixel (x,y) est en y*width+x. Il y a un plantage a la fin du programme car tu ne mets pas protect_leaves a true. Quand tu ecris whiteLeaf->protect_leaves, ca ne fait rien: c'est juste une variable, pas une methode.
FATOUX	Lambert	3	Tu ne dois pas definir les fonctions dans un .h. Le programme plante de temps en temps car tu oublies d'initialiser ta variable sontDesFeuilles a true. Tu ne traites pas les images rectangulaires avec des pixels "virtuels". Le taux de compression ?
FAUDUET	Alex	4	Bon ensemble. Code clair, mals quelques fuites memoire. Il aurait ete bien que le programme teste plusieurs seuils et affiche les taux de compression. La question 6 n'est pas correctement traitee, il faut prendre en compte la taille des structures.
FERNANDEZ	Raphaël	4.5	Petite fuite memoire: avant de faire delete [] DAG, il faut faire un delete DAG[i] pour i=0...255. A noter que protect_leaves est une variable statique, donc elle ne s'adresse pas a un objet mais a une classe (comme une variable globale dans un namespace), donc il suffit de faire QuadTree<byte>::protect_... =true pour que ca s'applique a tous. Il aurait ete bien de tester la compression avec un seuil adaptatif.
FRANCOIS	Quentin	4.5	Le taux de compression peut etre meilleur que ca: des QuadTree<byte> peuvent etre utilises, et les feuilles ne doivent pas etre comptees comme multiples car elles sont factorisees. Il faut eviter les variables globales. Essayer de faire un peu plus le menage avec des delete.

GAINON	Louise	3.5	Pour le taux de compression, distinguer QuadNode et QuadLeaf, qui n'ont pas la meme taille. Tu n'as pas fait l'image de taille virtuelle puissance de 2. Pas de tableau statique de taille variable, c'est interdit par le standard. Utiliser protect_leaves n'etait pas si complique.
GINOULHAC	Raphaël	4	Le code est un peu fastidieux, avec toutes ses variantes. Une fonction unique de compression et une de decompression aurait pu traiter tous les cas de figure. La variable globale n_leaves_dag est un peu malvenue, il y avait mieux a faire. NB: pour utiliser sizeof, pas besoin de creer un objet, elle prend en realite un type: sizeof(QLeaf<byte>).
HÉMADOU	Louis	4.5	C'est une tres mauvaise idee d'utiliser tes propres classes Image et ColorImage, qui n'ont pas de constructeur par copie. Tu as de la chance de ne pas provoquer de crash. En tout cas, il y a des fuites memoire. Il est dommage de tout reecrire pour les images couleur, il suffit de separer en trois images grises et de rassembler a la fin.
LAINÉE	Martin	1.5	Le programme plante car le tableau image2 n'est pas alloue avant remplissage dans Lecture. Il est inutile de recalculer a chaque appel recursif la puissance de 2 qui convient, elle ne change pas. Ne pas coder en dur dans le code la taille de l'image. Il manque des abs pour calculer des distances entre niveaux de gris.
LANDAIS	Jean	2	Les crashs ne sont pas dus a une fuite memoire (ce n'est jamais le cas, sauf a saturer la memoire, auquel cas on a une exception bad_alloc), mais a des lectures/ecritures hors de bornes: encodage_rect ne verifie pas que le pixel existe (ca peut etre un pixel virtuel hors image), decode_rect appelle decode_carre, etc.
LASRY	Raphaël	4.5	Tu ne reponds pas vraiment a la question 6: il faut distinguer les noeuds et les feuilles, qui ont des tailles differentes. De plus, on peut utiliser des QuadTree<byte> plutot que QuadTree<int>. A noter que protect_leaves est une variable globale qui ne s'applique pas a un objet particulier.

LÊ	Paul-Vinh	4	La variable seuil globale est une mauvaise idee, il n'y a aucune raison de ne pas en faire une variable locale. Un bug grave dans img_to_quad (non appele dans ton main !): delete T; return T qui peut avoir des consequences fatales au programme. Il aurait ete bien que le main teste toutes les images.
LE GRELLE	Hugues	4.5	Il est dommage d'avoir fait tant de fonctions differentes, alors que la methode generale (taille rect, niv gris) s'applique pour les cas particuliers. Quelques fuites memoire. Le main est bien, il teste plusieurs images et parametres, il est dommage que le taux de compression ne soit pas calcule. A noter que des QuadTree<byte> suffisent.
LESUEUR	Louis	4.5	Beaucoup de fuites memoire. Le taux de compression est completement faux : sizeof(quad_horse) est la taille d'un noeud, sizeof(horse) est la taille d'une image, qui ne comprend pas ses pixels (allocation dynamique). C'est bien d'avoir traite une image couleur et non carree. Il est bizarre que le constructeur de Image fasse un affichage.
LOISON	Virginie	4	Le code est mal organise: ce n'est pas dans quadtree.h et quadtree.cpp qu'il faut mettre les transformations image ↔ arbre. Il ne faut pas creer une image plus grande que necessaire, il faut considerer que quand on sort de l'image les pixels sont a 0. Il faut des abs pour des distances de niveau de gris.
MEDINA	François	4.5	Pour le taux de compression, il etait possible de prendre des QuadTree<byte>, dont les feuilles peuvent etre plus petites. Mais puisque tu utilises un DAG pour les feuilles, il ne faut pas les compter comme dupliquees. Il aurait ete interessant de comparer differents niveaux de compression. Petite fuite memoire: avant delete[] leaves, if faut des delete leaves[i]. Rattraper des exceptions n'est generalement pas considere comme un fonctionnement "normal" du programme. Il vaut mieux tester si ce sont des feuilles explicitement.
MOREL	Sébastien	4.5	Tu n'as pas compare la taille en memoire de l'arbre pour calculer le taux de compression. Pour les images rectangulaires, il etait demande de ne pas les agrandir, mais de completer par des pixels "virtuels" de valeur constante. Bien pour l'image couleur et pour les tests mutliples dans ton main.

OUHAÏCHI	Firas	4	Le taux de compression se calcule en prenant en compte la taille complete de l'arbre en memoire, il faut donc utiliser sizeof. L'argument seuil de encode_image_grey est inutilise, car le seuil est base sur seuil_ref.
PAN	Chao	4	Tu n'as pas vraiment compris le protect_leaves, qui sert en fait uniquement dans le cas DAG pour ne pas supprimer plusieurs fois les feuilles. Pour calculer le taux de compression, il faut prendre en compte la taille totale de l'arbre, et utiliser des sizeof. Dommage que ton main ne reproduise pas toutes les experiences du rapport.
PION	Aurélien	4	Il manque le taux de compression, qui depend de la taille totale de l'arbre et fait intervenir sizeof. Tu n'as pas vraiment compris l'interet de protect_leaves. Ca se voit par exemple avec quadtr_image_niveau, ou le noir est privilegie par rapport aux autres niveaux.
POLETTE	Nadège	5.5	Excellent travail ! Pour le taux de compression, il faut utiliser sizeof, et les noeuds et les feuilles n'ont pas la meme taille. A noter qu'il est aussi possible de faire un DAG en niveaux de gris, puisqu'il n'y a que 256 feuilles possibles.
POLI	Maxime	5.5	Bravo ! En fait, tu aurais pu compresser plus le fichier de l'arbre en ecrivant en binaire. Autre possibilite: compresser le fichier texte arbre.txt (j'obtiens un facteur 10). NB: protect_leaves est une variable statique de classe, c'est comme une variable globale dans un namespace. Ca ne s'adresse donc pas a un objet en particulier.
POUCIN	Florentin	4	Il etait dmande de ne pas agrandir les images pour les images rectangulaires, mais d'utiliser des pixels "virtuels". Tu ne calcules pas le taux de compression, il faut pour ca utiliser sizeof(QuadNode<int>). A noter que QuadTree<byte> convient aussi mieux que int.
QUILY	Ludovic	4	Les taux de compression ne sont pas calcules. Il faut pour ca utiliser sizeof. Il faut aussi s'efforcer de desallouer la memoire obtenue dynamiquement.
RIOU	Auriane	4	Il ne fallait pas creer une image plus grande si elle est rectangulaire, mais utiliser des pixels "virtuels". De plus, ta fonction adapts_dimension est peu pratique car elle ne dit pas si elle a fait de l'allocation ou pas (meme si ca peut se verifier en comparant le pointeur resultat avec l'entree). Il faut aussi liberer les arbres construits.

RUGET	Simon	4	Ton programme ne calcule pas les taux de compression. Dommage que les 2/3 de ton main soient en commentaire, a priori c'est du code qui devait marcher. La variable protect_leaves permet de ne pas faire de delete sur les feuilles, ce qui est prudent car les feuilles sont partagees et on ne veut pas faire plusieurs delete sur le meme pointeur.
SANCHEZ CRUZ	Jenny		
SCHLEGEL	Nicolas	3.5	Le calcul de taux de compression est faux. Sizeof image est equivalent a sizeof(byte*), c'est-a-dire la taille d'un pointeur (8 octets). La taille d'une feuille de l'arbre n'est pas 1, mais sizeof(QuadLeaf<int>). Ton main fait de l'allocation statique de tableau avec une taille variable, c'est interdit par le standard. Il faut s'efforcer de liberer la memoire allouee par new.
SOUSSI	Nada	4.5	Pour le taux de compression, il faut utiliser sizeof pour avoir la taille totale de l'arbre. C'est bien d'avoir voulu liberer la memoire, mais il faut faire attention: utiliser delete[] pour un tableau, delete a,b ne libere que a et pas b (il faut des delete separes), et tu as besoin de protect_leaves a true pour ne pas liberer deux fois la meme feuille partagee dans le DAG (plantage a la fin du programme chez moi).
STÄMPFLI	Erwan	3.5	Quand tu partages les feuilles avec un DAG, si les 4 sous-regions sont des feuilles avec la meme intensite, il ne faut pas creer une nouvelle feuille, mais renvoyer FeuilleBlanche ou FeuilleNoire suivant cette intensite. Tu n'as pas compris l'interet de protect_leaves. Il faut les delete dans le main. Tu n'as pas calcule le taux de compression, il faut utiliser sizeof pour ca.
TRINH	Robin	1.5	Le programme plante des le premier calcul d'arbre. C'est du en partie au parametre height passe dans le main pour le y de BuildQuadtree. Mais meme en corrigeant en 0, il reste un bug.
VAN DEN BERGH	Candice	4.5	Il y a un bug memoire avec dessin_carres (ecriture en-dehors des bornes du tableau) qui fait planter le programme (sur ma machine) avant le premier affichage. Dommage, car c'est bien sinon. Attention de bien faire delete[] pour les tableaux.

VENARD	Paul-Louis	4	Le taux de compression n’est pas tout a fait exact, car les noeuds et les feuilles ne prennent pas la meme taille memoire, et il faut distinguer le cas DAG et arbre. Ta fonction de decodage ne supporte que des images carrees de taille puissance de 2, ce qui est une limitation. Il manque des delete pour les arbres. Pour le cas gris, meler DAG et arbre n’est pas une tres bonne idee, car comment va-t-on detruire les feuilles autres que lwhite et lblack ?
VINCENT	Théo	3.5	Tu n’as pas compris a quoi servait protect_leaves, qui est une variable globale qui evite de faire un delete pour une feuille. Son but est d’eviter un double delete quand on fait un DAG, ce que tu ne fais pas. Ton encodage ne marche que pour les images carrees de taille puissance de 2. Le taux de compression n’est pas exact, il faut faire sizeof pour avoir la taille d’un objet.
VONGPASEUT	Clarine	4.5	Bon travail. Pour les liberations memoire, attention de regler protect_leaves, qui doit etre a true pour le DAG et a false sinon.

SetHash(/5)

ALLAIN	Clément	5	TB !
BAKONG EPOUNÉ	Killian	4.5	Pour les histogrammes, count/insert n'est pas tres efficace, operator[] est bien mieux car on n'a pas besoin de verifier l'existence de la cle avant. Il aurait fallu mesurer le temps de calcul et comparer a la methode naïve.
BEL KHAYAT ZOUGGARI	Yassine	4.5	Ton calcul des isotoponymes me semble avoir un bug. Tu n'entres jamais dans FindSameNameIter, et heureusement car celle-ci va provoquer une erreur memoire grave: comme son argument S n'est pas passe par reference, tu retournes des iterateurs sur un ensemble temporaire qui n'existera plus en sortie.
BERTRAND	Romain	4	Il y a un bug dans Town::operator< car seul le nom est compare. Cela perturbe les isotoponymes. Tu ne mesures pas les temps de calcul et tu n'implementes pas la methode naïve.
BIRAC	Nicolas	3.5	Il y a un bug dans le redimonsionnement du vector hist (fonction histogram): il faut au moins it->second.size()+1 elements. Tu n'as pas implemente la methode naïve pour comparer les temps de calcul. De plus, tu peux grandement accelerer en remontant des if plus haut dans l'imbrication des for.
BOUGHDIRI	Ahmed	4.5	Le gain de temps de error_cities aurait ete plus spectaculaire si tu avais remonte les if que tu pouvais de la meme facon que error_cities_naive. Les conditions des for avec des virgules n'ont aucun sens, seul le premier terme compte. L'operateur virgule ne signifie ni "et" ni "ou". De toutes facons, ca aurait ete faux d'arreter la boucle quand une des conditions etait verifiee.
BRAMI	Victoria	3	La deuxieme partie de tes fonctions histogramme_ est plus complexe que necessaire, bien que correcte. Dommage que tu n'aies pas fait les dernieres questions.
BRISINGER	Maxime	3.5	Tes fonctions ensemble_villes_meme sont inefficaces, le for interne est inutile car les villes concernees seront atteintes plus tard par l'iterateur it. Meler iterateurs et operator[] dans la fonctions confusions est etrange. Il faut etre consistant, et preferer les iterateurs. Tu ne mesures pas le temps de calcul et ne compares pas avec la methode plus intelligente.

CANCES	Adrien	4.5	On peut faire mieux dans nb_towns_possibly_mistaken, on n'a pas besoin de towns, seul le sous-ensemble I compte. Il manque la version naïve et la comparaison des temps de calcul. La fonction histogram avec count/insert peut etre amelioree avec map::operator[]. La tableau retourne par ces fonctions n'est jamais utilise.
CHAKIR	Adel	4	Tes fonctions same_name_neighbor_ sont illisibles a cause de noms de variables iterateurs trop longs. L'approche naïve doit avoir un probleme, elle ne devrait pas etre si lente meme sur le jeu complet de villes. De plus, elle detecte des doublons, alors qu'il n'y en a pas.
CHAMBON	Loick	3.5	count/insert est inefficace pour remplir l'histogramme. L'operateur [] est bien mieux, il fonctionne meme pour de nouvelles cles. La fonction nom_identique est lente aussi, car towns_with_name parcourt tout le vecteur. La version inefficace des isotoponymes est fausse car les conditions sur les id sont dans le for, ce qui provoque un arret de boucle premature.
CHANCEREL	Pia	4.5	Dans ta version naïve, les conditions du type j!=NC.end() sont inutiles car j!=i va s'assurer qu'on ne peut pas depasser I. Mais la vraie version naïve itere sur le vecteur towns, pas sur NC. On peut cependant l'accelerer en remontant des conditions du if plus haut dans les boucles for.
COSSON	Margot	5	Tres bon travail, rien a redire !
DESHORS	Charles	4.5	Curieusement, ton histogramme des villes de meme position est faux car tu comptes par exemple 4 fois le bin 4 alors que ca ne correspond qu'a une paire de coordonnees. Tu n'as pas fait cette erreur dans l'histogramme des noms. Pour remplir la map, find/insert n'est pas le mieux a faire, operator[] est plus puissant. A part ca, le reste est bien.
DESJARDINS	Antoine	3.5	Il est interdit de faire un tableau statique de taille variable, en l'occurrence towns.size(). Compter sur le fait que les villes soient triees par nom est tricher. Pour l'histogramme des positions, il y a un bug memoire dans l'agrandissement de liste2. Ca provoque des resultats faux chez moi, mais ca pourrait aussi bien faire planter le programme. Il manque le code de la version naïve et la mesure du temps de calcul empirique.

DURIF	Anne	2.5	J'ai eu bcp de mal a faire compiler le code: variable non definie (car dans la partie commentee), probleme de const, etc. Town::operator< est incorrect car il n'utilise que le nom. Le resultat de histoCoor est faux car Point2D::operator< n'est pas un ordre total. Ton utilisation de la constante nb est dangereux, car tu ne verifies pas que l'histogramme ne depasse pas.
FATOUX	Lambert	3.5	count/insert pour remplir la map n'est pas optimal, il vaut mieux utiliser operator[]. Les histogrammes sont bons, mais il y a certainement une erreur dans les isotoponymes: la methode intelligente est plus rapide et la methode naïve plus lente. Comparer les id dnas la condition du for est une erreur, car on arrete des qu'ils coincident.
FAUDUET	Alex	4	Utiliser try/catch pour faire un test ordinaire c'est detourner son usage et c'est plus lent qu'un test direct. En fait, operator[] aurait convenu parfatiement, sans if. Il faudrait mesurer le temps de l'approche proposee et implementer la version naïve pour comparer.
FERNANDEZ	Raphaël	4.5	La version efficace des isotoponymes doit iterer sur N inter C. Du coup, le temps de calcul est largement reduit, d'un facteur 200 chez moi en mode Release. Attention de passer les vector par reference aux fonctions pour eviter une copie. Bon travail par ailleurs.
FRANCOIS	Quentin	3.5	La plupart des fonctionnalites demandees ne sont pas executees car en commentaire. histogramme_coo n'affiche pas l'histogramme. Les temps de calcul ne sont pas mesures. Il faut passer les vector par reference aux fonctions pour eviter des copies inutiles. Pour une map, inutile d'inserer les valeurs, c'est automatique avec l'operator [].
GAINON	Louise	4.5	La procedure pour les histogrammes est un peu compliquee a suivre mais fonctionne. Il aurait ete bien de mesurer le temps de calcul et de comparer avec la recherche exhaustive. Passer les vector en const& pour eviter les copies.
GINOULHAC	Raphaël	5	Bien. Il aurait fallu mesurer les temps de calcul empiriques pour verifier la borne theorique. La methode naïve est plus bete que ca, elle fait les boucles sur toutes les villes.

HÉMADOU	Louis	4	Le code est bien ecrit, mais il y a une erreur dans le nombre des isotoponymes: si v1 et v4 ont le meme nom, n1=n4 et N1C4 contient v4 et N4C1 contient v1, ce qui ne suffit pas pour une ambiguite... En fait, tu veux que v1 et v4 n'aient ni meme nom ni meme position. Il aurait aussi fallu mesurer le temps de calcul.
LAINÉE	Martin	1.5	Il etait demande de ne pas utiliser find. La fonction calculCN est bien plus complexe que necessaire, il suffisait d'utiliser set_intersection, et elle a un bug car le nombre d'elements est faux. Il manque le calcul des isotoponymes.
LANDAIS	Jean	5	Dans le calcul final des isotoponymes, sortir les if le plus possible des for permet d'aller bcp plus vite, et meme la methode naïve tourne en quelques secondes en mode Release.
LASRY	Raphaël	4.5	Pour les histogrammes, il y a un probleme dans la formulation de l'affichage et un nom de villes utilise par n villes doit incrementer le bin n de l'histogramme de 1 et pas de n. Bien pour le reste. Bien passer les vector et set en const& pour eviter des copies.
LÊ	Paul-Vinh	4.5	Dans les histogrammes, il est assez maladroit de diviser par (i+1) a la fin, il suffisait d'incrementer de 1 au lieu de i+1. Il aurait ete bien de mesurer le temps empirique par les deux methodes des isotoponymes. Le rapport est bien. Passer les vector en const& aux fonctions.
LE GRELLE	Hugues	4.5	Ton set_intersection ne marchait pas car ton ensemble resultat inser est vide. Or set_intersection ne fait pas des insert pour ajouter les elements pertinents, il se contente d'ecrire dans un OutputIterator. Il aurait fallu mesurer les temps empiriques de calcul pour les deux methodes d'isotoponymes. Passer les vector en const& aux fonctions.
LESUEUR	Louis	4.5	Il y a une erreur de raisonnement dans le calcul des isotoponymes, il n'y en a pas 69. Regarder par exemple ce qui se passe avec deux villes differentes de meme nom dans it et itt. En fait, tu as besoin queit et itt ne partagent ni le meme nom ni la meme position.
LOISON	Virginie	5	La methode naïve que tu proposes n'est pas si naïve, car elle utilise inter et non towns. Bien passer par const& les conteneurs aux fonctions.

MEDINA	François	5	TB. La methode naïve peut quand meme etre grandement acceleree en remontant les if aussi haut que possible apres le for. La methode vraiment naïve serait d'iterer sur towns et non sur NinterC, mais en remontant les if elle tourne encore en temps acceptable.
MOREL	Sébastien	4	Tu n'utilises pas assez la librairie standard du C++: pour faire le tri (qui etait inutile en fait), il faut utiliser sort. Pour l'intersection de deux ensembles, il faut utiliser set_intersection. Tu aurais du comparer les temps de calcul mesures avec la methode naïve. Bon rapport.
OUHAÏCHI	Firas	1	Code correct, mais retard trop important dans le rendu.
PAN	Chao	3.5	Il etait demande de ne pas utiliser find. Le code est meme beaucoup plus simple sans l'utiliser. Tu aurais du comparer le temps de calcul a celui de la methode naïve des isotoponymes.
PION	Aurélien	3.5	Les valeurs trouvees sont fausses car operator< de Town n'utilise pas le nom de la ville. Revois la definition d'un histogramme. Le code est inutilement complique par endroits, avec par exemple des passages comme (ret.first)->second.second, code tres difficile a comprendre. Passer les conteneurs par reference aux fonctions.
POLETTE	Nadège	5	Tres bon travail, mais la manipulation des map peut se faire beaucoup plus simplement: faire count[it->name()]++ suffit, meme si it->name() n'a jamais ete vu avant.
POLI	Maxime	5	TB. Curieusement, le premier if de ton tromper_naif commencait par if(v1!=v1 au lieu de if(v1!=v2, d'ou un temps de calcul quasi nul.
POUCIN	Florentin	3	Ta fonction error_town correspond a la version naïve. La version plus rapide iterer sur l'ensemble N inter C. Mais le test des id dans le for est une erreur, ca veut dire qu'on arrete la boucle des que id est retrouve, au lieu de juste sauter cette occurrence. Il y a mieux que find/insert pour une map avec l'operateur []. Il faut passer les conteneurs par reference aux fonctions pour eviter la copie.

QUILY	Ludovic	3	Pas de tableau statique avec nb variable d'elements, c'est interdit par le standard ! Inutile de faire une passe pour initialiser a 0 une map, c'est automatique avec operator[]. L'approche naïve peut etre largement plus rapide en remontant les if tant que possible. L'approche intelligente peut se faire bcp plus simplement en utilisant inter au lieu de towns dans l'approche naïve.
RIOU	Auriane	4	Il est un peu bizarre de passer par une map<string,Point2D> pour en fait ne jamais utiliser la valeur Point2D. Il est plus direct d'utiliser map<string,int> ou map<Town,int>. Tu n'implementes pas et ne mesures pas le temps de la methode naïve. Attention de passer les vector en const& pour eviter les copies.
RUGET	Simon	4	Construire les histogrammes en complexite N^2 est franchement inefficace. De plus, ton parcours du tableau homographe pour chacune des x possibles de l'histogramme est aussi inefficace. Attention de passer les vector et set en const& aux fonctions.
SANCHEZ CRUZ	Jenny		
SCHLEGEL	Nicolas	4.5	Il est dommage que tu n'implementes pas la methode naïve, qui itere sur towns plutot que N_inter_C, pour comparer les temps de calcul. Bien par ailleurs.
SOUSSI	Nada	4.5	Bon travail. Il est simplement dommage que tu ne mesures pas les temps de calcul et que tu n'implementes pas la methode naïve. A noter que reserver towns.size() pour l'intersection est largement surevalue, puisque la taille finale est majoree par min(|N|,|C|).
STÄMPFLI	Erwan	2	Ton bug est du a une erreur bete dans Point2D::operator<, ton else if etant equivalent au if. Une fois cela corrige, l'histogramme de C est correct. Cela corrige, il reste plusieurs bugs par la suite que tu n'as pas pu constater bien sur. Parmi ceux-ci: NinterC.begin()-t est negatif, il faut inverser les operandes. Pour que set_intersection fonctionne, il faut que les ensembles soient ordonnes suivant la meme relation d'ordre. Or ceux-ci sont des vector, remplis suivant des ordres differents.

TRINH	Robin	3	find/insert est inefficace pour une map, il faut utiliser operator[]. Ton CardC est faux car s'il y a 1083 noms de villes repetes 2 fois, ca fait 2*1083 villes, etc. Ton operator< de Town est incomplet puisqu'il ne compare que le nom. Du coup, N2 et C2 n'ont pas le bon nombre d'elements. Il aurait fallu implementer aussi la methode naïve et comparer les temps de calcul observes.
VAN DEN BERGH	Candice	4.5	C'est tres bien, mais tu signales en commentaire qu'on gagne beaucoup en iterant sur NinterC, sans le tester dans ton code. C'est dommage, ca aurait ete parfait.
VENARD	Paul-Louis	2	OK pour les histogrammes. Pour la suite, c'est Town::operator< qui cause tous les problemes. L'ordre lexicographique n'est pas name1<name2 && p1<p2, le second terme doit etre (name1==name2 && p1<p2). Ta boucle d'isotoponymes est terriblement inefficace: pour chaque it de NxC, tu dois parcourir toutes les villes au lieu de celles de NxC dans vect_v2 et vect_v3.
VINCENT	Théo	4.5	Dommage que tu n'implementes pas la methode naïve pour pouvoir comparer les temps de calcul empiriquement.
VONGPASEUT	Clarine	4	Il aurait fallu implementer la methode naïve et comparer les temps de calcul. Il est inutile de passer le nombre d'elements d'un vector en parametre, la methode size() permet de la retrouver. Il faut mieux utiliser un iterateur pour parcourir un vector. Bien passer les maps, vector et set en const& pour eviter les copies.