LA CITÉ
LE COLLÈGE D'ARTS APPLIQUÉS ET DE TECHNOLOGIE

PLAN DE COURS

Titre du cours: Assurance de qualité logicielle
Cote du cours: 025914
Durée du cours: 42 heures
Programme: Technologie du génie informatique

Description

À la fin de ce cours, l'étudiant sera en mesure de contribuer aux processus, pratiques et activités d'assurance qualité du logiciel et de l'amélioration du logiciel selon les normes et modèles en vigueur.

Pondération

Unité d'apprentissage	Pondération (%)
025947 - Documentation logicielle, bonnes pratiques en programmation	35 %
025948 - Diagnostiques de compilation, techniques de débogage	35 %
025949 - Gestion de configurations et de versions logicielles	30 %
TOTAL:	100 %

Système de notation

La note de passage de ce cours est de 60%.

A+ 90-100	B+ 77-79	C+ 67-69	DR 55-59
A 85-89	B 74-76	C 64-66	EC 0-54
A- 80-84	B- 70-73	C- 60-63	

Division en unités d'apprentissage

Unité d'apprentissage

Titre du cours: Documentation logicielle, bonnes pratiques en programmation
Cote du cours: 025947
Durée du cours: 14 heures
Programme: Technologie du génie informatique

Description

Ce module présente les pratiques Agile éprouvées de standardisation de code source, incluant les principes, patrons et pratiques de rédaction et de documentation, ainsi que les techniques et outils de refactorisation facilitant la mise en application de ces pratiques.

Compétences du programme ciblées par ce cours

- Participer à la conception, l'implantation et la mise en oeuvre de tests visant à assurer la qualité et la conformité des systèmes informatiques;
- Déterminer, analyser et documenter les besoins d'un client dans le but d'en extraire des exigences claires et applicables;

Indicateurs de développement

À la fin de ce cours, l'étudiant qui réussit sera en mesure de:

- Appliquer les bonnes pratiques générales visant à assurer la qualité d'un produit logiciel, et documenter adéquatement ses éléments afin d'en assurer sa compréhension et sa maintenabilité.

Évaluations

Évaluations formatives

- **Étude de cas - Renommage d'identificateurs**
  L'apprenant doit réviser un code source distribué en appliquant les principes et stratégies de nommage de variables, de fonctions et de classes étudiées en classe.
  Le but de l'évaluation est de sensibiliser l'apprenant au lien cause à effet entre une stratégie de nommage d'identificateurs et la maintenance d'un code source au sein d'une équipe de programmeurs.
- **Projet individuel - Concevoir une classe**
  L'apprenant doit concevoir une classe selon des spécifications fournies tout en appliquant les principes de rédaction étudiés en classe.
  L'instructeur fournit à l'apprenant une liste de spécifications imposant des fonctionnalités à implanter sous forme de classe (programmation orientée objet), ainsi que des tests unitaires validant ces fonctionnalités, et l'apprenant doit implanter la classe correspondante.
  Le but de l'évaluation est la mise en pratique des principales stratégies de rédaction de classes étudiées en classe.
- **Étude de cas - Documenter un code source**
  L'apprenant doit appliquer les principes de documentation étudiés en classe au code source produit par un pair lors de l'évaluation formative précédente.
  Le but de l'évaluation est d'amener l'apprenant à saisir l'importance de nommer adéquatement les identificateurs dans un code source (car il doit comprendre et documenter un code rédigé par autrui), ainsi qu'à l'importance de bien documenter le code source.
- **Étude de cas - Formater un code source**
  L'apprenant doit réviser le code source de la classe qu'il a rédigé lors de l'évaluation formative #3 (et documentée par autrui lors de l'évaluation formative #4) afin d'en formater le code selon les principes étudiés en classe.
  Le but de l'évaluation est de simuler un contexte d'entreprise où l'apprenant doit produire du code selon un standard de formatage imposé par l'employeur.
  Un outil de validation de formatage tel que StyleCop ou Checkstyle doit être appliqué par l'apprenant pour valider son code résultant.
- **Projet individuel - Abstraction de données**
  L'apprenant doit restructurer la classe résultant de l'évaluation formative précédente pour satisfaire des spécifications imposées.
  L'instructeur fournit une classe de base à partir de laquelle l'apprenant doit dériver sa classe. Pour valider la classe résultante, l'instructeur fournit à l'apprenant des tests unitaires pour valider son travail.

Le but de l'évaluation est d'amener l'apprenant à saisir l'importance de bien structurer le code source d'une classe lorsque celle-ci doit être intégrée dans projet d'équipe.

- **Projet individuel - Refactorisation**
  L'apprenant doit réviser un code source distribué en appliquant les principes et stratégies de refactorisation étudiées en classe.
  Le but de l'évaluation est d'amener l'apprenant à se familiariser avec les outils de refactorisation disponibles dans un environnement de développement intégré.

Évaluation(s) sommative(s)

- **Bonnes pratiques en programmation**
  L'apprenant doit réviser un code source imposé afin de les rendre conforme aux bonnes pratiques étudiées en classe.
  L'évaluation est divisée en deux parties : une première partie évalue les connaissances terminologiques de l'apprenant à l'aide de cartes mémoire. La seconde partie de l'évaluation est pratique et demande à l'apprenant d'exploiter ses notions acquises pour réviser, restructurer et documenter un code source fourni.
  Le but de l'évaluation est de faire une synthèse des bonnes pratiques étudiées et de les mettre en pratique.

Système de notation

La note de passage de ce cours est de 60%.

A+ 90-100	B+ 77-79	C+ 67-69	DR 55-59
A 85-89	B 74-76	C 64-66	EC 0-54
A- 80-84	B- 70-73	C- 60-63	

Contenu

- Sensibilisation à l'importance de la standardisation du code source dans un projet d'envergure afin d'en assurer la qualité, la maintenance et l'évolution.
- Principes de nommage et d'organisation de fichiers de code source, conventions de nommage d'identificateurs, Recommandations sur l'écriture des instructions, des structures de contrôle et l'usage des parenthèses dans les expressions, et bonnes pratiques de rédaction de fonctions.
- Principes de documentation de code source et générateurs de documentation, styles rédactionnels, de formatage et d'indentation.
- Conception de classes assurant l'abstraction de données afin de supporter les principes de la programmation orientée objet (héritage, encapsulation et polymorphisme).
- Revue et mises en pratique des principales stratégies de refactorisation de code source, incluant le renommage, l'encapsulation de champs, l'extraction de fonctions, l'extraction d'interfaces, et la réorganisation et suppression de paramètres de fonctions.

Modalités pédagogiques

- Études de cas
- Discussions
- Cartes mémoire
- Projets individuels

Séquence pédagogique

Modules	Lectures, travaux et évaluations
L'importance d'un code source propre et soigné Conséquences du mauvais code Règles et principes de base	Discussions
Organisation et nommage de fichiers de code source Nommage d'identificateurs : variables, fonctions, classes, interfaces et domaines	Études de cas Évaluation formative #1

Modules	Lectures, travaux et évaluations
Rédaction de fonctions : objectifs, nomenclatures et paramètres Particularités des fonctions membres : accès aux données, fonctions statiques, encapsulation Paramètres d'entrée, de sortie et d'entrée/sortie Gestion des erreurs via les code d'erreurs et les exceptions	Projet individuel Évaluation formative #2
Documentation de code source : pertinence, bons commentaires, mauvais commentaires Commentaires balisés Outils de génération de documentation	Études de cas Évaluation formative #3
Principes de formatage de code source Formatage vertical (densité, distance et ordonnancement) Formatage horizontal (alignement, indentation, blocs)	Études de cas Évaluation formative #4
Objets et structures de données abstraites Importance de l'encapsulation des données Accessibilités aux membres Propriétés Membres statiques Interfaces	Projet individuel Évaluation formative #5
Survol des principes et stratégies de refactorisation Renommage Encapsulation de champs Extraction de fonctions Extraction d'interfaces Réorganisation de paramètres de fonctions Suppression de paramètres de fonctions	Projet individuel Cartes mémoire Évaluation formative #6 Évaluation sommative

Exigences

Exigences obligatoires

- L'étudiant est responsable de prendre connaissance de son Guide de programme disponible dans le Portail étudiant. Il y trouvera notamment les directives de l'étudiant fournissant des informations précises en ce qui a trait à son cheminement scolaire, l'intégrité scolaire, les règles de conduite, les équivalences et reconnaissances des acquis, l'ajout et l'abandon de cours, le transfert de programme et les évaluations de reprise en cas d'échec.
- L'étudiant a la responsabilité d'être présent au cours, de participer et d'apporter tout outil, document ou fourniture exigé.
- L'étudiant dont le comportement est jugé inapproprié peut se voir refuser l'accès au cours.
- L'intégrité scolaire fait partie intégrante de l'apprentissage de l'étudiant. Elle vise à promouvoir un environnement où l'honnêteté intellectuelle occupe une place de choix. Le plagiat et la fraude scolaire constituent un manquement à l'intégrité scolaire. Toute forme de plagiat et de fraude scolaire est interdite et sera sanctionnée selon les dispositions indiquées dans les Directives pédagogiques du Collège.
- Tous les devoirs et les travaux doivent être remis à la date indiquée.
- La présentation des devoirs et des travaux doit se conformer aux exigences indiquées dans le cours.

Exigences particulières

- SO

Cohérence pédagogique

Résultats d'apprentissage de la formation professionnelle (RAFP)

Ce cours contribue à l'atteinte des connaissances, aptitudes et attitudes suivantes :

- Identifier, analyser, concevoir, développer, mettre en oeuvre, vérifier et documenter les exigences reliées au contexte de l'informatique.
- Diagnostiquer, dépanner, document et surveiller les problèmes techniques.
- Développer, tester et maintenir des applications logicielles pour l'intégration de systèmes.

Résultats d'apprentissage relatifs à l'employabilité (RARE)

Ce cours contribue à l’atteinte des connaissances, habiletés et attitudes suivantes qui sont essentielles à la réussite professionnelle ainsi qu’à l’apprentissage continu :

- Communiquer d'une façon claire, concise et correcte, sous la forme écrite, orale et visuelle, en fonction des besoins de l'auditoire (no.1).

Unité d'apprentissage

Titre du cours: Diagnostiques de compilation, techniques de débogage
Cote du cours: 025948
Durée du cours: 14 heures
Programme: Technologie du génie informatique

Description

Ce module présente des techniques pour diagnostiquer les messages d'erreur et d'avertissement liés à la compilation d'un code source, ainsi que les divers outils de débogage permettant de diagnostiquer et corriger les erreurs logiques dans un code source.

Compétences du programme ciblées par ce cours

- Participer à la conception, l'implantation et la mise en oeuvre de tests visant à assurer la qualité et la conformité des systèmes informatiques;

Indicateurs de développement

À la fin de ce cours, l'étudiant qui réussit sera en mesure de:

- Expliquer les causes et solutions aux avertissements et erreurs de syntaxe les plus communes, et appliquer des techniques de débogage, incluant la journalisation et le profilage, pour identifier et corriger les erreurs logiques les plus communes.

Évaluations

Évaluations formatives

- **Étude de cas - Corriger des erreurs**
  L'apprenant doit analyser des messages d'erreur et d'avertissement produits lors de la compilation d'un code source fourni et proposer des solutions afin de les circoncire.
  Le but de l'évaluation est de permettre à l'apprenant de se familiariser avec les messages de diagnostiques les plus fréquemment produits lors de la compilation d'un code source.
- **Étude de cas -Déboguer un code à l'aide du traçage**
  L'apprenant doit exploiter des stratégies de traçage (telles que le print et le assert) pour diagnostiquer des comportements erronés d'un code source à l'exécution.
  Le but de l'évaluation est d'amener l'apprenant à identifier les avantages et limites du traçage pour déboguer un code source.
- **Étude de cas - Déboguer une classe orientée objet**
  L'apprenant doit diagnostiquer des erreurs syntaxiques et logiques spécifiques à un code source orienté objets.
  Le but de l'évaluation est d'initier l'apprenant aux difficultés liées au débogage d'un code orienté objets exploitant l'héritage, la surcharge et le polymorphisme.
- **Étude de cas - Déboguer un code source**
  L'apprenant doit exploiter un débogueur pour localiser des erreurs logiques dans un code source.
  Le but de l'évaluation est de familiariser l'apprenant aux fonctionnalités communes aux outils de débogages généralement intégrés aux environnement de développement, tels que le point d'arrêt, l'exécution pas à pas et l'inspection des variables.

Évaluation(s) sommative(s)

- **Identifier des erreurs dans un code source**
  L'apprenant doit exploiter les outils de traçage et de débogage pour diagnostiquer et corriger des erreurs syntaxiques et logiques dans divers codes sources.
  L'évaluation est divisée en deux parties: une première partie évalue à l'aide de cartes mémoire les connaissances de l'apprenant sur la cause d'erreurs de syntaxe rencontrées couramment en programmation, et une seconde partie consiste en une mise en situation où on demande à l'apprenant d'utiliser des outils de traçage et de débogage pour corriger des erreurs logiques dans un code source imposé.
  Le but de l'évaluation est de faire une synthèse des notions de débogage étudiées et de les mettre en pratique.

Système de notation

La note de passage de ce cours est de 60%.

A+ 90-100	B+ 77-79	C+ 67-69	DR 55-59
A 85-89	B 74-76	C 64-66	EC 0-54
A- 80-84	B- 70-73	C- 60-63	

Contenu

- Analyse et correction d'erreurs de syntaxe et de sémantique dans un code source, ainsi que les principaux messages d'avertissements liés à la conformité du code source aux standards du langage de programmation.
- Techniques de traçage pour fins de débogage d'un code source, incluant le traçage d'appels de fonctions et d'évolution de données, ainsi que l'utilisation des assertions.
- Stratégies de débogage adaptées à la programmation orientée objets telles que le traçage d'invocations de méthodes via pointeurs polymorphiques et l'exploitation de références statiques versus dynamiques.
- Débogueurs intégrés à un environnement de développement pour diagnostiquer des erreurs logiques dans un code source, avec survol des principales fonctionnalités : points d'arrêt, exécution pas à pas, inspection de variables et suivi de pile d'appels.

Modalités pédagogiques

- Études de cas
- Cartes mémoire
- Discussions

Séquence pédagogique

Modules	Lectures, travaux et évaluations
Objectifs du module Motivations des aptitudes de débogage Stratégies de débogage	Discussions
Erreurs courantes de syntaxe Messages d'avertissements fréquents Erreurs de sémantique	Études de cas Évaluation formative #1
Fondements du traçage d'exécution Erreurs typiques identifiables via le traçage Utilisation d'indicateurs de traçage Traçage via invocations de fonctions Traçage d'évolution de données Utilisation de la macro d'assertion	Études de cas Évaluation formative #2
Traçage de chaînes de caractères Traçage de pointeurs Traçage de structures et de classes	Études de cas Évaluation formative #3
Survol des fonctionnalités d'un débogueur intégré Les points d'arrêt inconditionnels et conditionnels L'exécution pas à pas et ses spécificités Inspection de contenus de variables Inspection de la pile d'appel	Études de cas Évaluation formative #4
Ckecklist de la détection et de prévention d'erreurs Techniques de test fondamentaux applicables	Cartes mémoire Évaluation sommative

Exigences

Exigences obligatoires

- L'étudiant est responsable de prendre connaissance de son Guide de programme disponible dans le Portail étudiant. Il y trouvera notamment les directives de l'étudiant fournissant des informations précises en ce qui a trait à son cheminement scolaire, l'intégrité scolaire, les règles de conduite, les équivalences et reconnaissances des acquis, l'ajout et l'abandon de cours, le transfert de programme et les évaluations de reprise en cas d'échec.
- L'étudiant a la responsabilité d'être présent au cours, de participer et d'apporter tout outil, document ou fourniture exigé.
- L'étudiant dont le comportement est jugé inapproprié peut se voir refuser l'accès au cours.
- L'intégrité scolaire fait partie intégrante de l'apprentissage de l'étudiant. Elle vise à promouvoir un environnement où l'honnêteté intellectuelle occupe une place de choix. Le plagiat et la fraude scolaire constituent un manquement à l'intégrité scolaire. Toute forme de plagiat et de fraude scolaire est interdite et sera sanctionnée selon les dispositions indiquées dans les Directives pédagogiques du Collège.
- Tous les devoirs et les travaux doivent être remis à la date indiquée.
- La présentation des devoirs et des travaux doit se conformer aux exigences indiquées dans le cours.

Exigences particulières

- SO

Cohérence pédagogique

Résultats d'apprentissage de la formation professionnelle (RAFP)

Ce cours contribue à l'atteinte des connaissances, aptitudes et attitudes suivantes :

- Identifier, analyser, concevoir, développer, mettre en oeuvre, vérifier et documenter les exigences reliées au contexte de l'informatique.
- Diagnostiquer, dépanner, document et surveiller les problèmes techniques.
- Développer, tester et maintenir des applications logicielles pour l'intégration de systèmes.

Résultats d'apprentissage relatifs à l'employabilité (RARE)

Ce cours contribue à l’atteinte des connaissances, habiletés et attitudes suivantes qui sont essentielles à la réussite professionnelle ainsi qu’à l’apprentissage continu :

- Appliquer une approche systématique de résolution de problèmes (no.5).
- Utiliser une variété de stratégies pour prévoir et résoudre des problèmes (no.6).

Unité d'apprentissage

Titre du cours: Gestion de configurations et de versions logicielles
Cote du cours: 025949
Durée du cours: 14 heures
Programme: Technologie du génie informatique

Description

Un système de gestion de versions (SGV) est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus, permettant notamment de retrouver les différentes versions d'un lot de fichiers connexes. Ils sont utilisés notamment en développement logiciel pour conserver le code source relatif aux différentes versions d'un logiciel.
Ce module présente les caractéristiques communes à tous les systèmes de gestion de versions, et initie l'apprenant à collaborer à des projets de développement logiciel collaboratifs gérés par SGV, soient en ligne de commande ou via un environnement de développement intégré.

Compétences du programme ciblées par ce cours

- Participer à l'analyse, à la conception et au développement des systèmes informatiques, les implanter et en assurer la maintenance;

Indicateurs de développement

À la fin de ce cours, l'étudiant qui réussit sera en mesure de:

- Utiliser un système de versions décentralisé dans un contexte de travail en équipe afin de conserver le code source relatif aux différentes versions d'un logiciel.

Évaluations

Évaluations formatives

- **Projet individuel - Premiers pas avec un SGV**
  L'apprenant doit installer un système de gestion de versions (SGV) en ligne de commande sur un poste de travail local et explorer ses fonctionnalités.
  Le but de l'évaluation est d'initier l'apprenant à ce qu'est un SGV, ses caractéristiques et fonctionnalités. Le fait d'exploiter un SGV en ligne de commande assure une versatilité de l'apprenant dans l'utilisation des SGV.
- **Projet individuel - Créer et configurer un dépôt**
  L'apprenant doit créer et configurer un nouveau dépôt de projet dans Git à l'aide de GitHub.
  Le but de l'évaluation est de permettre à l'apprenant de se familiariser avec le processus de création d'un dépôt, incluant son processus d'invitation à la collaboration.
- **Projet d'équipe - Collaborer à un dépôt**
  L'apprenant doit collaborer à un dépôt existant afin d'apporter des modifications à son code source.
  Le but de l'évaluation est d"initier l'apprenant aux multiples étapes menant à la modification (branchements) et l'acceptation (commits) de celles-ci, incluant la gestion de conflits potentiels (issues).
- **Projet d'équipe -Utiliser un SGV intégré à un EDI**
  L'apprenant doit participer à un projet collaboratif via l'environnement de développement intégré (EDI) IntelliJ.
  Le but de l'évaluation est de permettre à l'apprenant de se familiariser avec l'utilisation d'un SGV via une interface graphique plutôt qu'en ligne de commande.

Évaluation(s) sommative(s)

- **Utilisation d'un SGV pour collaborer**
  L'apprenant doit exploiter un SGV pour contribuer à un dépôt de code source partagé avec ses pairs.
  L'évaluation est divisée en deux parties :une première partie évalue à l'aide de cartes mémoire la maîtrise des notions théoriques acquises par l'apprenant. La seconde partie de l'évaluation est pratique et demande à l'apprenant d'exploiter ses notions acquises pour collaborer à un dépôt de code source existant.
  Le but de l'évaluation est de faire une synthèse des interactions d'un programmeur avec un SGV et de les mettre en pratique afin de contribuer à

un projet collaboratif.

Système de notation

La note de passage de ce cours est de 60%.

A+ 90-100	B+ 77-79	C+ 67-69	DR 55-59
A 85-89	B 74-76	C 64-66	EC 0-54
A- 80-84	B- 70-73	C- 60-63	

Contenu

- Introduction aux systèmes de gestion de versions (SGV) logicielles : leurs raisons d'être, leur utilité, les concepts clés et inventaire des principaux SGV exploités en industrie.
- Présentation du système de gestion de versions Git et de son interface en ligne de commande GitHub : installation et première utilisation individuelle. Créer un dépôt de projet, y déposer du code et récupérer des modifications, et visualiser les opérations.
- Éditer un projet via un système de gestion de versions (SGV) : branchements, modifications et gestion de répertoires.
- Collaborer à un projet via un système de gestion de versions (SGV) : gérer des collaborations, effectuer des commits, participer aux requêtes de déchargements et gérer les conflits.
- Inventaire des systèmes de gestion de versions (SGV) retrouvés dans des environnements de développement intégrés (EDI) : IntelliJ, Eclipse et Visual Studio.

Modalités pédagogiques

- Discussions
- Projets individuels
- Projets d'équipe
- Cartes mémoire

Séquence pédagogique

Modules	Lectures, travaux et évaluations
Objectifs du module Concepts clés Inventaire des solutions SGV	Discussions
Présentation de GitHub Comparer Git aux autres solutions SGV Installer Git Faire un premier commit Historique des opérations Reculer à un commit spécifique	Projet individuel Évaluation formative #1
Présentation des remotes GitHub, qu'est-ce que c'est ? Récupérer du code d'un autre dépôt Créer un dépôt Déposer du code sur GitHub Récupérer des modifications	Projet individuel Évaluation formative #2
Créer des branches Fusionner des branches Résoudre un conflit Identifier l'auteur d'une modification Ignorer des fichiers Éviter des commits superflus Contribuer à des projets open source	Projet d"équipe Évaluation formative #3

Modules	Lectures, travaux et évaluations
Systèmes SGV intégrés IntelliJ Eclipse Visual Studio	Projet d'équipe Cartes mémoire Évaluation formative #4 Évaluation sommative

Exigences

Exigences obligatoires

- L'étudiant est responsable de prendre connaissance de son Guide de programme disponible dans le Portail étudiant. Il y trouvera notamment les directives de l'étudiant fournissant des informations précises en ce qui a trait à son cheminement scolaire, l'intégrité scolaire, les règles de conduite, les équivalences et reconnaissances des acquis, l'ajout et l'abandon de cours, le transfert de programme et les évaluations de reprise en cas d'échec.
- L'étudiant a la responsabilité d'être présent au cours, de participer et d'apporter tout outil, document ou fourniture exigé.
- L'étudiant dont le comportement est jugé inapproprié peut se voir refuser l'accès au cours.
- L'intégrité scolaire fait partie intégrante de l'apprentissage de l'étudiant. Elle vise à promouvoir un environnement où l'honnêteté intellectuelle occupe une place de choix. Le plagiat et la fraude scolaire constituent un manquement à l'intégrité scolaire. Toute forme de plagiat et de fraude scolaire est interdite et sera sanctionnée selon les dispositions indiquées dans les Directives pédagogiques du Collège.
- Tous les devoirs et les travaux doivent être remis à la date indiquée.
- La présentation des devoirs et des travaux doit se conformer aux exigences indiquées dans le cours.

Exigences particulières

- SO

Cohérence pédagogique

Résultats d'apprentissage de la formation professionnelle (RAFP)

Ce cours contribue à l'atteinte des connaissances, aptitudes et attitudes suivantes :

- Appliquer les principes et les outils de gestion de projet lors du travail sur des projets dans le secteur de l'informatique.
- Développer, tester et maintenir des applications logicielles pour l'intégration de systèmes.

Résultats d'apprentissage relatifs à l'employabilité (RARE)

Ce cours contribue à l’atteinte des connaissances, habiletés et attitudes suivantes qui sont essentielles à la réussite professionnelle ainsi qu’à l’apprentissage continu :

- Appliquer une approche systématique de résolution de problèmes (no.5).
- Utiliser une variété de stratégies pour prévoir et résoudre des problèmes (no.6).

Service d’appui et d’adaptation

Les étudiants qui ont ou qui croient avoir une ou des limitations fonctionnelles permanentes ou temporaires qui nécessitent des mesures d’accommodement académiques sont priés de s’inscrire le plus tôt possible auprès du Service La Boussole, situé au local C1030. Un document provenant d’un professionnel de la santé réglementé décrivant la nature et la portée des limitations fonctionnelles est requis pour l’obtention des mesures d’accommodement académiques, mais un étudiant pourrait recevoir des mesures d’accommodement intérimaires dans l’attente de cette documentation. Tous les documents provenant des professionnels de la santé réglementés seront conservés de façon confidentielle à La Boussole et la copie originale sera rendue. Pour plus d’information, veuillez contacter le (613) 742-2483 poste 2090 ou par courriel à laboussole@collegelacite.ca

Reconnaissance des acquis non scolaires

Les étudiants intéressés à se prévaloir du processus de reconnaissance des acquis peuvent obtenir des renseignements relatifs au processus d’évaluation en consultant le Guide de votre programme.