Total /50

Identification

Prénom: .....

Nom: .....

Code permanent: .....

Instructions

- L'examen dure 3 heures
- Vous avez droit à une feuille de notes recto-verso, au format lettre ($8\ 1/2 \times 11$)
- Avant de quitter, vous devez présenter votre carte étudiante et signer la feuille de présence
- Pour chacune des questions à choix multiples, vous devez sélectionner exactement une réponse, en utilisant une croix ($\times$) ou un crochet ($\checkmark$)
- Pour chacune des questions, vous pouvez supposer que:
  - `sizeof(int)` retourne 4;
  - `sizeof(char)` retourne 1;
  - `sizeof(bool)` retourne 1 et
  - `sizeof(double)` retourne 8 et
  - `sizeof(T*)` retourne 8, peu importe `T`.

Considérez le programme C suivant:

```
#include <stdio.h>
struct S {
  int i;
  int* j;
};
union U {
  int k;
  int* m;
};
int main(void) {
  struct S s[2];
  union U* u[2];
  printf("%ld %ld %ld %ld\n",
         sizeof(struct S), sizeof(s),
         sizeof(union U), sizeof(u));
  return 0;
}
```

Q1. /2 Qu'est-ce qui sera affiché sur la sortie standard lors de son exécution?

(a) Valeur 1: .....**12 ou 16** .....

(b) Valeur 2: .....**24 ou 32** .....

(c) Valeur 3: .....**8** .....

(d) Valeur 4: ..... **16** .....

Q2. /4 (Suite de la question précédente) En vous référant au programme précédent, indiquez à l'aide d'une croix ($\times$) ou d'un crochet ($\checkmark$), pour chacune des expressions suivantes, si elle peut être utilisée comme valeur gauche (*lvalue*) et/ou comme valeur droite (*rvalue*). *Note*: chaque ligne peut contenir 0 crochet, 1 crochet ou 2 crochets.

Expression	*lvalue*	*rvalue*
`&(s[1].j)`		$\times$
`*(s + 1)`	$\times$	$\times$
`&&u`		
`u[0]->m + 1`		$\times$

Q3. /2 Considérez le programme C suivant:

```
#include <stdbool.h>
#include <stdio.h>
void f(int* a, unsigned int n,
       bool (*p)(int), void (*g)(int)) {
  for (unsigned int i = 0; i < n; ++i)
    if (p(a[i]))
      g(a[i]);
}
void g(int i) {
  printf("%d ", i);
}
bool p(int i) {
  return i % 2 == 0;
}
int main(void) {
  int a[] = {1, 4, 1, 4, 2, 1};
  f(a, 6, p, g);
}
```

Qu'est-ce qui sera affiché sur la sortie standard lors de son exécution?

.....**4 4 2** .....

Q4. /2 (Suite de la question précédente) Proposez des noms significatifs pour les fonctions `f`, `g` et `p`:

(a) `f`: .....`appliquer_action_si` .....

(b) `g`: .....`afficher_entier` .....

(c) `p`: .....`est_pair` .....

Q5. /2 Considérez le programme C suivant:

```
#include <stdlib.h>
int main(void) {
  int* p = malloc(sizeof(int));
  unsigned int n = 1;
  do {
    n *= 2;
    p = realloc(p, n * sizeof(int));
  } while (p != NULL);
  free(p);
  return 0;
};
```

Pour chacune des affirmations suivantes, indiquez si elle est vraie ou fausse en encerclant la lettre appropriée (V ou F). Le programme...

(a) a un comportement indéterminé. V / (F)

(b) entraîne une fuite de mémoire. (V) / F

(c) accède à de la mémoire interdite. V / (F)

(d) génère une erreur de segmentation. V / (F)

Q6. /1 On souhaite implémenter le type abstrait "ensemble d'entiers" de sorte que les opérations d'insertion et de suppression soient le plus efficace possible. Parmi les implémentations suivantes, quelle est théoriquement la plus adaptée?

- ☐ Tableau statique trié
- ☐ Tableau dynamique trié
- ☐ Liste simplement chaînée
- ■ Arbre binaire de recherche équilibré

Q7. /2 Considérez le programme C suivant:

```
#include <stdio.h>
#define MAX(X, Y) X > Y ? X : Y
#define MINUS(X, Y) X - Y
int main(void) {
  int x = -2, y = 3;
  printf("%d\n", MAX(x - y, x + y));
  printf("%d\n", MINUS(x - y, y - x));
  return 0;
}
```

Qu'est-ce qui sera affiché sur la sortie standard lors de son exécution?

.....**1 et -6** .....

Q8. /2 Considérez le programme C suivant:

```
#include <stdarg.h>
#include <stdbool.h>
#include <stdio.h>
int a(int n, ...) {
  va_list list;
  va_start(list, n);
  int j = -1;
  for (unsigned int i = 0;
       j == -1 && i < n;
       ++i) {
    if (!va_arg(list, int))
      j = i;
  }
  va_end(list);
  return j;
}
int main(void) {
  printf("%d\n", a(4, 3, 2, 1, 0));
}
```

Qu'est-ce qui sera affiché sur la sortie standard lors de son exécution?

.....**3** .....

Q9. /4 Pour chacune des situations suivantes, indiquez s'il s'agit d'un problème de compilation (C), d'édition des liens (L), d'utilisation (U), de justesse (J) ou de robustesse (R). Choisissez une seule réponse par situation.

(a) L'outil GCC indique ne pas trouver le fichier `cairo.h` .....**C** .....

(b) L'outil GCC indique qu'un symbole est défini plusieurs fois .....**L** .....

(c) Le programme arrête son exécution en raison d'un manque de mémoire .....**U/J/R** .....

(d) Le programme déclenche une erreur de segmentation sur GitLab seulement ....**R** ....

(e) Le programme retourne des valeurs différentes selon l'environnement ..... **R** .....

Q10. /1 Quelle option de GCC permet de définir un symbole à la compilation?

■ `-D` ☐ `-M` ☐ `-S` ☐ `-X`

Q11. /1 En programmation, quel est le nom de la stratégie qui préconise de cacher les détails d'implémentation à l'utilisateur?

☐ Cohésion ■ Encapsulation
☐ Couplage ☐ Modularité

Q12. /2 Considérez le Makefile suivant:

```
.PHONY: build clean

CFLAGS = -Wall -Wextra
exec = e
modules = a b c
modules_c_files = $(patsubst %,%.c,$(modules))
modules_h_files = $(patsubst %,%.h,$(modules))
modules_o_files = $(patsubst %,%.o,$(modules))

build: $(exec)

$(exec): $(exec).o $(modules_o_files)
  gcc $^ -o $@

$(exec).o: $(exec).c
  gcc $(CFLAGS) -c $<

$(modules_o_files): %.o: %.c %.h
  gcc $(CFLAGS) -c $<

clean:
  rm -f $(modules_o_files)
  rm -f $(exec)
```

Supposons que le répertoire courant contienne ce Makefile, ainsi que les fichiers suivants:

```
$ ls
Makefile   a.h    b.h     c.h
a.c        b.c    c.c     e.c
```

Donnez la liste de tous les fichiers qui seront produits si on entre la commande `make` (et que celle-ci réussit).

.....`a.o, b.o, c.o, e.o, e` .....

Q13. /1 (Suite de la question précédente) Parmi les variables définies plus haut, laquelle devrait-on modifier si on souhaite compiler le projet avec les fichiers d'en-tête de la bibliothèque Jansson?

..... `CFLAGS` .....

Q14. /1 (Suite de la question précédente) Donnez la liste de tous les fichiers qui seront supprimés si on entre la commande `make clean` (en supposant avoir déjà entré la commande `make` plus tôt).

.....`a.o, b.o, c.o, e` .....

Q15. /6 Considérez la fonction `max` définie comme suit:

```
int max(unsigned int n, int* a) {
  int m = a[0];
  for (unsigned int i = 1; i < n; ++i)
    m = m > a[i] ? m : a[i];
  return m;
}
```

Quelle est sa complexité cyclomatique? ....**3** ....

Proposez un cadre de tests pour la fonction `max` qui couvre tous les branchements possibles de façon minimale, en remplissant le tableau suivant (le nombre de tests doit être égal à la complexité cyclomatique):

Test	`n`	`a`
1	`1`	`{1}`
2	`2`	`{1, 2}`
3	`2`	`{2, 1}`

Q16. /2 Pour chacune des affirmations suivantes, indiquez si elle est vraie (V) ou fausse (F).

(a) Certaines options d'optimisation de GCC ne respectent pas le standard (V) / F

(b) Certaines options de GCC permettent de réduire la taille de l'exécutable (V) / F

(c) Pour optimiser un programme, il faut d'abord se concentrer sur son design (V) / F

(d) L'optimisation d'un programme peut avoir un impact sur sa maintenance (V) / F

Q17. /15 Considérez le programme incomplet suivant:

```
#include <assert.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NB_MAX_VILLES 100
#define NB_MAX_ROUTES 100

struct Ville {                          // Une ville
  char* id;                             // L'identifiant de la ville
};

struct Route {                          // Une route
  char* id;                             // L'identifiant de la route
  const struct Ville* ville1;           // Les deux villes reliées par la route
  const struct Ville* ville2;
  unsigned int longueur;                // La longueur de la route (en km)
  unsigned int vitesse_max;             // La vitesse maximale permise sur la
                                        // route (en km/h)
};

struct Carte {                          // Une carte
  unsigned int nb_villes;               // Le nombre de villes dans la carte
  struct Ville villes[NB_MAX_VILLES]; // Les villes de la carte
  unsigned int nb_routes;               // Le nombre de routes dans la carte
  struct Route routes[NB_MAX_ROUTES]; // Les routes de la carte
};

/**
 * Initialise une carte
 * @param carte  La carte à initialiser
 */
void initialiser_carte(struct Carte* carte) {
  // TODO 1
}

/**
 * Ajoute une ville à une carte
 * Note: si la capacité maximale de la carte est atteinte, la ville n'est pas
 * ajoutée et le programme continue son exécution normalement.
 * @param carte  La carte
 * @param id     L'identifiant de la ville
 */
void ajouter_ville(struct Carte* carte, const char* id) {
  assert(carte != NULL);
  assert(id != NULL);
  if (carte->nb_villes == NB_MAX_VILLES) return;
  carte->villes[carte->nb_villes].id = strdup(id);
  ++carte->nb_villes;
}

/**
```

```
 * Retourne l'indice d'une ville selon son identifiant
 * Note: si l'identifiant n'est pas trouvé, retourne -1.
 * @param carte  La carte
 * @param id     L'identifiant de la ville
 */
int indice_ville(const struct Carte* carte, const char* id) {
  assert(carte != NULL);
  assert(id != NULL);
  for (unsigned int v = 0; v < carte->nb_villes; ++v)
    if (strcmp(carte->villes[v].id, id) == 0)
      return v;
  return -1;
}

/**
 * Ajoute une route à une carte
 * Note: si la capacité maximale de la carte est atteinte, la route n'est pas
 * ajoutée et le programme continue son exécution normalement.
 * @param carte        La carte
 * @param id           L'identifiant de la route
 * @param id1          L'identifiant de la première ville
 * @param id2          L'identifiant de la deuxième ville
 * @param longueur     La longueur de la route
 * @param vitesse_max  La vitesse maximum permise sur la route
 */
void ajouter_route(struct Carte* carte,
                   const char* id,
                   const char* id1,
                   const char* id2,
                   unsigned int longueur,
                   unsigned int vitesse_max) {
  // TODO 2
}

/**
 * Affiche une carte sur stdout
 * @param carte  La carte à afficher
 */
void afficher_carte(const struct Carte* carte) {
  assert(carte != NULL);
  printf("Villes\n");
  for (unsigned int v = 0; v < carte->nb_villes; ++v)
    printf("  %s\n", carte->villes[v].id);
  printf("Routes\n");
  for (unsigned int r = 0; r < carte->nb_routes; ++r)
    printf("  %s\n", carte->routes[r].id);
}

/**
 * Détruit une carte
 * @param carte  La carte à détruire
 */
void detruire_carte(struct Carte* carte) {
  // TODO 3
}
```

```c
/**
 * Retourne la durée minimale d'un trajet sur une route
 * @param carte   La carte
 * @param id      L'identifiant de la route
 * @return        La durée du trajet (en heures)
 */
double duree_minimale_trajet(const struct Carte* carte,
                             const char* id) {
  // TODO 4
}

int main(void) {
  struct Carte carte;
  initialiser_carte(&carte);
  ajouter_ville(&carte, "gatineau");
  ajouter_ville(&carte, "montreal");
  ajouter_ville(&carte, "quebec");
  ajouter_ville(&carte, "sherbrooke");
  ajouter_route(&carte, "mtl-gat", "montreal", "gatineau", 200, 110);
  ajouter_route(&carte, "mtl-qc", "montreal", "quebec", 250, 120);
  ajouter_route(&carte, "mtl-sh", "montreal", "sherbrooke", 150, 120);
  ajouter_route(&carte, "qc-sh", "quebec", "sherbrooke", 230, 100);
  afficher_carte(&carte);
  printf("Durée minimale MTL->GAT: %lf\n",
         duree_minimale_trajet(&carte, "mtl-gat"));
  printf("Durée minimale MTL->QC: %lf\n",
         duree_minimale_trajet(&carte, "mtl-qc"));
  detruire_carte(&carte);
}
```

Complétez les fonctions `initialiser_carte` (`// TODO 1`), `ajouter_route` (`// TODO 2`), `detruire_carte` (`// TODO 3`) et `duree_minimale_trajet` (`// TODO 4`) de telle sorte qu'elles respectent le comportement décrit dans les *docstrings*. En particulier, on s'attend à ce que le programme affiche ceci sur la sortie standard lors de l'exécution:

```
Villes
  gatineau
  montreal
  quebec
  sherbrooke
Routes
  mtl-gat
  mtl-qc
  mtl-sh
  qc-sh
Durée minimale MTL->GAT: 1.818182
Durée minimale MTL->QC: 2.083333
```

De plus, vous devez utiliser la programmation par contrats pour gérer les erreurs, en insérant des assertions appropriées pour valider les préconditions suivantes:

- L'argument `carte` ne doit jamais être `NULL`
- Les identifiants des villes et des routes ne doivent jamais être `NULL`
- Dans `ajouter_route`, lorsqu'on réfère à l'identifiant d'une ville, celui-ci doit exister dans la carte

- Dans `duree_minimale_trajet`, lorsqu'on réfère à l'identifiant d'une route, celui-ci doit exister dans la carte

Note: La lisibilité et l'efficacité de votre solution seront prises en considération dans l'évaluation.

```
// TODO 1:
void initialiser_carte(struct Carte* carte) {
  assert(carte != NULL);
  carte->nb_villes = 0;
  carte->nb_routes = 0;
}
```

```
// TODO 2:
void ajouter_route(struct Carte* carte,
                   const char* id,
                   const char* id1,
                   const char* id2,
                   unsigned int longueur,
                   unsigned int vitesse_max) {
  assert(carte != NULL);
  assert(id != NULL);
  assert(id1 != NULL);
  assert(id2 != NULL);
  int v1 = indice_ville(carte, id1);
  assert(v1 != -1);
  int v2 = indice_ville(carte, id2);
  assert(v2 != -1);
  if (carte->nb_routes == NB_MAX_ROUTES) return;
  struct Route* route = carte->routes + carte->nb_routes;
  route->id = strdup(id);
  route->ville1 = carte->villes + v1;
  route->ville1 = carte->villes + v2;
  route->longueur = longueur;
  route->vitesse_max = vitesse_max;
  ++carte->nb_routes;
}
```

```
// TODO 3:
void detruire_carte(struct Carte* carte) {
  assert(carte != NULL);
  for (unsigned int v = 0; v < carte->nb_villes; ++v)
    free(carte->villes[v].id);
  for (unsigned int r = 0; r < carte->nb_routes; ++r)
    free(carte->routes[r].id);
}
```

```
// TODO 4:
double duree_minimale_trajet(const struct Carte* carte,
                             const char* id) {
  assert(carte != NULL);
  assert(id != NULL);
  for (unsigned int r = 0; r < carte->nb_routes; ++r) {
    const struct Route* route = carte->routes + r;
    if (strcmp(route->id, id) == 0)
      return (double)route->longueur / (double)route->vitesse_max;
  }
  assert(false);
}
```