Algorithmique et Programmation – Examen (durée 1h30)

Première session du 21 juin 2024

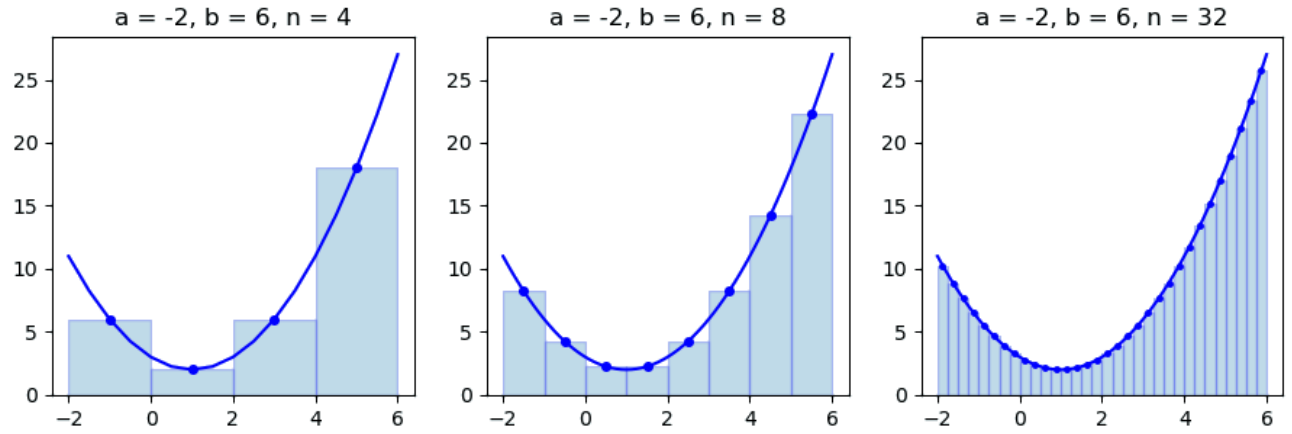
Notes : Aucun document autorisé. Sont interdits les calculatrices, les téléphones, ainsi que tout autre ustensile de calcul ou de communication.

Exercice 1 : Questions de connaissances générales (3 points)

6 questions

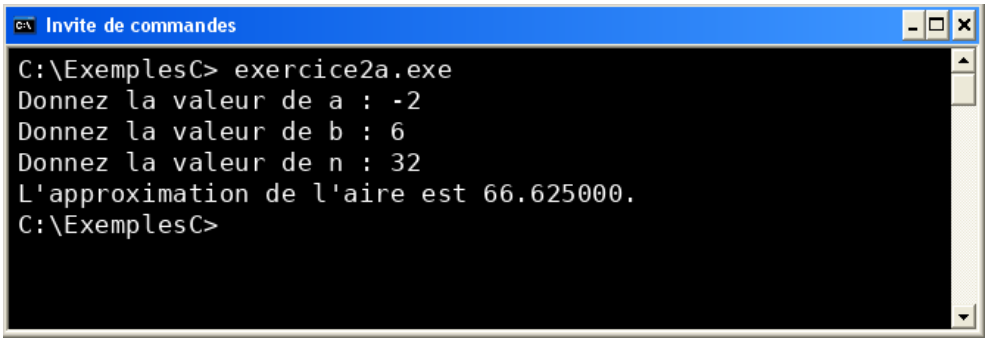
Exercice 2 : Deux programmes à compléter (7 points)

1) Nous voulons calculer la somme de Riemann [1] par méthode du point médian à pas constant pour estimer l'aire sous la courbe d'une fonction continue sur un intervalle $[a, b]$ en la subdivisant en n sous-intervalles de mêmes longueurs ($dx = (b - a)/n$). Par exemple, pour la fonction $f(x) = x^2 - 2x + 3$ sur l'intervalle $[-2, 6]$, pour $n = 4$, $n = 8$ et $n = 16$:



Cette méthode consiste donc à sommer l'aire des rectangles sur chaque sous-intervalle $[x_{i-1}, x_i]$ pour i de 1 à n d'une hauteur définie au point médian du sous-intervalle ($t_i = (x_{i-1} + x_i)/2$), sachant que $x_i = a$ et $x_n = b$. Par exemple, pour $n = 4$, la somme de Riemann vaut dans le cas présent $f(-1)*dx+f(1)*dx+f(3)*dx+f(5)*dx = (6 + 2 + 6 + 18) * 2 = 64$. La précision de l'estimation augmente avec n. Par exemple, la somme vaut 66 lorsque $n = 8$ et 66,6250 lorsque $n = 32$ et 66,656250 lorsque $n = 64$.

Complétez ci-dessous le programme, pour faire ce calcul en langage C pour la fonction donnée en exemple, dont voici un exemple d'exécution :



1. Établie par Bernhard Riemann, mathématicien allemand du XIXe siècle, dans son mémoire d'habilitation en 1854, puis publiée en 1867. La longue histoire de cette somme est retracée dans l'article suivant : Jean-Paul Truc. 2019. Riemann Sums for Generalized Integrals. *The College Mathematics Journal* 50, 2 (March 2019), 123–132. https://doi.org/10.1080/07468342.2019.1560119

```
_______ <_______>

double f(double x) _
  ______ x * x - 2 * x + 3_
_

___ ____() _
  double a, b, xi, yi, mrsum = 0, dx, ti_
  ___ i, n_

  ______("Donnez la valeur de a : ")_
  _____("%lf", &a)_

  ______("Donnez la valeur de b : ")_
  _____("%lf", &b)_

  ______("Donnez la valeur de n : ")_
  _____("%d", &n)_

  dx = (b - a) / n_

  ___ (i = 1_ i <= n_ i++) _
    xi = a + i * dx_
    ti = xi - dx * 0.5_
    yi = f(ti)_
    mrsum += yi * dx_
  _

  ______("L'approximation de l'aire est %lf.\n", mrsum)_

  ______ 0_
_
```

2) Nous souhaitons un programme permettant de calculer un coût de livraison à partir des informations contenues dans un fichier : numéro de bon de livraison, largeur, hauteur et profondeur du colis (en cm) et enfin sa masse (en kg). La catégorie tarifaire est déterminée en fonction des dimensions du colis : catégorie A si la somme excède 150 cm et catégorie B sinon. Le coût d'expédition dépend de la catégorie et de la masse du colis :

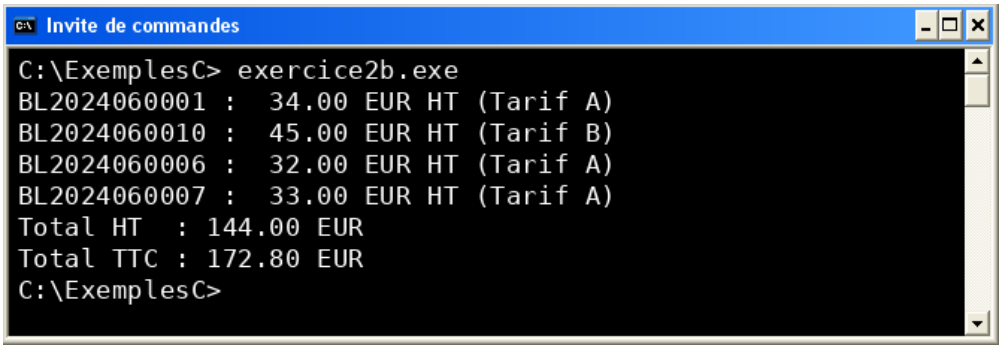
- Le tarif en catégorie A est de 30 euros HT plus 1 euro par unité de kilogramme entâmée (càd, le coût d'un colis de 3,6 kg est 30 + 4 = 34 euros).
- Le tarif en catégorie B est de 10 euros HT plus 5 euros par unité de kilogramme entâmée (càd, le coût d'un colis de 0,6 kg est 10 + 5 = 15 euros).

Complétez ci-dessous le programme en langage C pour calculer le coût total d'une livraison à partir du fichier suivant :

livraison.txt

```
BL2024060001 40 50 100 3.400
BL2024060010 30 30 70 6.500
BL2024060006 40 50 70 1.950
BL2024060007 40 50 70 2.600
```

Voici un exemple d'exécution du programme :



Remarque : dans le code du programme, les montants sont représentés par des entiers et expérimés en centimes d'euros.

```
________ <_______>

char classification(___ largeur, ___ hauteur, ___ profondeur) _
  ______ (largeur + hauteur + profondeur > 150) ? 'A' : 'B'_
_

___ tarification(char classe, float masse) _
  __ (classe == 'A') _
    ______ 3000 + 100 * ((___)(masse) + 1)_
  _
  ____ _
    ______ 1000 + 500 * ((___)(masse) + 1)_
  _
_

___ ____() _
  FILE *f_
  ___ L, H, P, cout, sum_
  float M_
  char numero[64], C_

  f = fopen("livraison.txt", "r")_

  while (!feof(f)) _
    __ (f_____(f, "%s %d %d %d %f", numero, &L, &H, &P, &M) == 5) _
      C = classification(L, H, P)_
      cout = tarification(C, M)_
      ______("%s : %6.2f EUR HT (Tarif %c)\n", numero, cout / 100.0, C)_
      sum += cout_
    _
  _

  fclose(f)_

  ______("Total HT  : %6.2f EUR\n", sum / 100.0)_
  ______("Total TTC : %6.2f EUR\n", (sum * (1 + 20 / 100.0)) / 100.0)_

  ______ 0_
_
```

Exercice 3 : Une fonction et deux programmes à compléter (10 points)

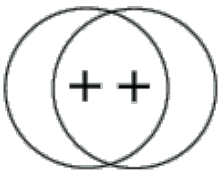
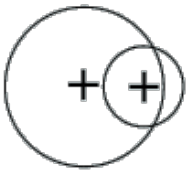
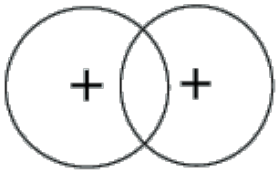
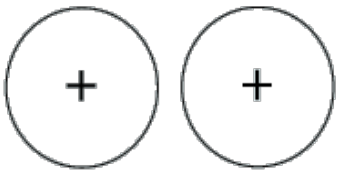
Les objets communicants sont de plus en plus nombreux (téléphones et consoles de jeux, casques de vélo, plantes vertes, ...). Nous nous intéressons à l'atteignabilité d'un objet par un autre pour savoir si les deux objets peuvent communiquer par ondes radio (sans nous préoccuper de la technologie utilisée). Pour simplifier, dans le cadre du présent exercice, nous considérons que les récepteurs ont tous la même sensibilité, qu'il n'y a pas d'obstacles entre émetteurs et récepteurs, et que la portée d'un émetteur est la distance maximum à laquelle un récepteur pourra réceptionner et démoduler le signal.

Le but des trois questions de cet exercice est d'initier une cartographie d'objets communicants afin de pouvoir calculer les communications possibles.

a) Pour commencer, Nous écrivons une fonction en langage C qui prend en paramètre :

1. c1x, c1y, c1r : le centre et le rayon d'un premier cercle ;
2. c2x, c2y, c2r : le centre et le rayon d'un deuxième cercle ;

et retourne le nombre entier :

3	2	1	0
Lorsque deux cercles incluent le centre de l'autre, ils sont en double inclusion.	Lorsque seulement un de deux cercles inclue le centre de l'autre, ils sont en simple inclusion.	Lorsque les deux cercles s'intersectent sans y inclure leurs centres, ils sont en intersection.	Lorsque deux cercles ne s'intersectent pas, ils sont en exclusion.
			
Càd, la distance euclidienne entre les centres des deux cercles est plus petite que les deux rayons des cercles.	Càd, la distance euclidienne entre les centres des deux cercles est plus petite que le rayon d'uniquement un des deux cercles.	Càd, la somme des deux rayons des cercles est plus grande que la distance euclidienne entre les centres des deux cercles.	Càd, la somme des deux rayons des cercles est plus petite que la distance euclidienne entre les centres des deux cercles.

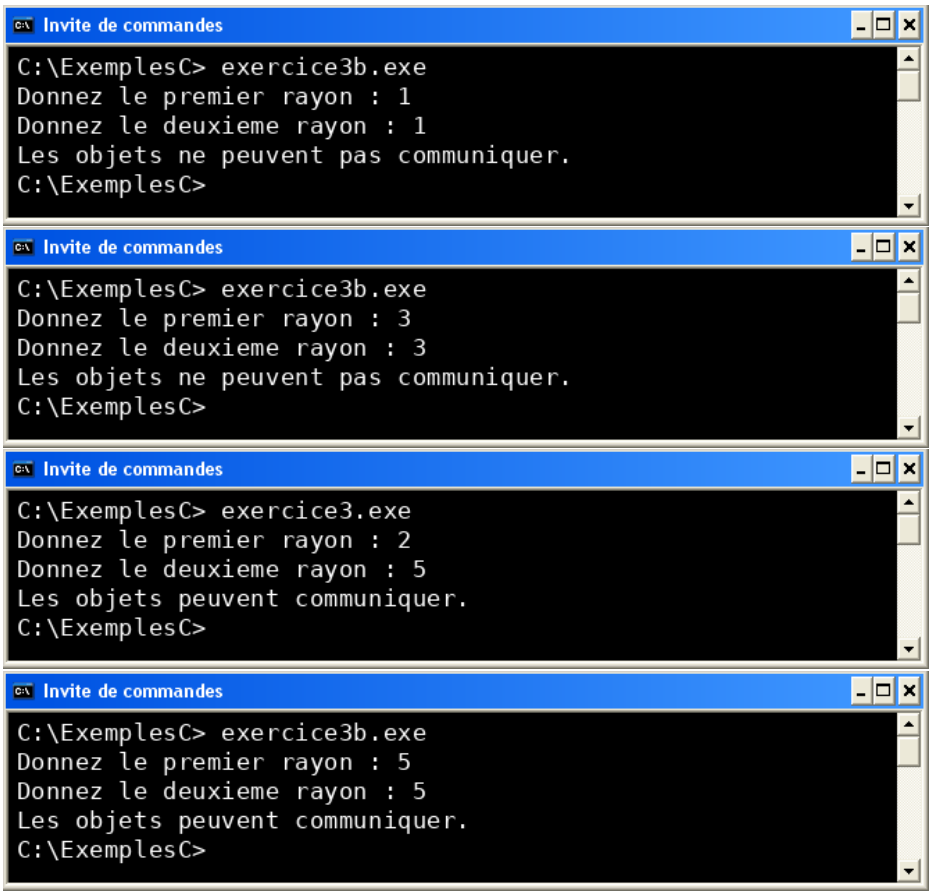
Le prototype de la fonction sera le suivant :

```
char inclusion_cercles (int c1x, int c1y, int c1r, int c2x, int c2y, int c2r) ;
```

Écrivez la déclaration de votre fonction, puis vérifiez son exécution par un programme de tests (déjà écrit).

b) Pour tester la fonction du (a), nous écrivons maintenant un programme en langage C qui vérifiera si la communication est possible entre deux objets de positions (-2, 3) et (2, 3) pour lesquels l'utilisateur indiquera le rayon de leurs portées respectives. Le programme indiquera alors si une communication est possible (bi-directionnelle, cas 3, ou uni-directionnelle, cas 2) ou impossible (cas 1 ou cas 0).

Ci-après quatre exemples d'exécution de ce programme :

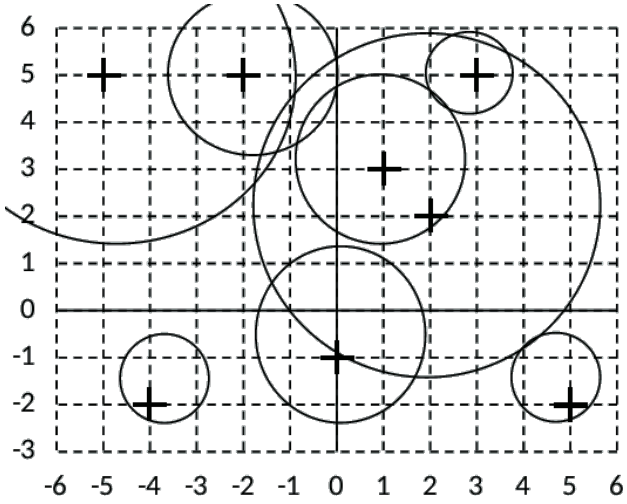


Écrivez le code du programme, puis vérifiez son exécution (une version déjà prête de la fonction sera ajoutée automatiquement lors du test, donc vous écrivez ici uniquement le programme demandé).

c) Pour amorcer le développement du logiciel de calcul, nous écrivons maintenant un un programme en langage C dans lequel sont déclarés les trois tableaux suivants, définissant 8 objets communicants et le rayon de leur portée :

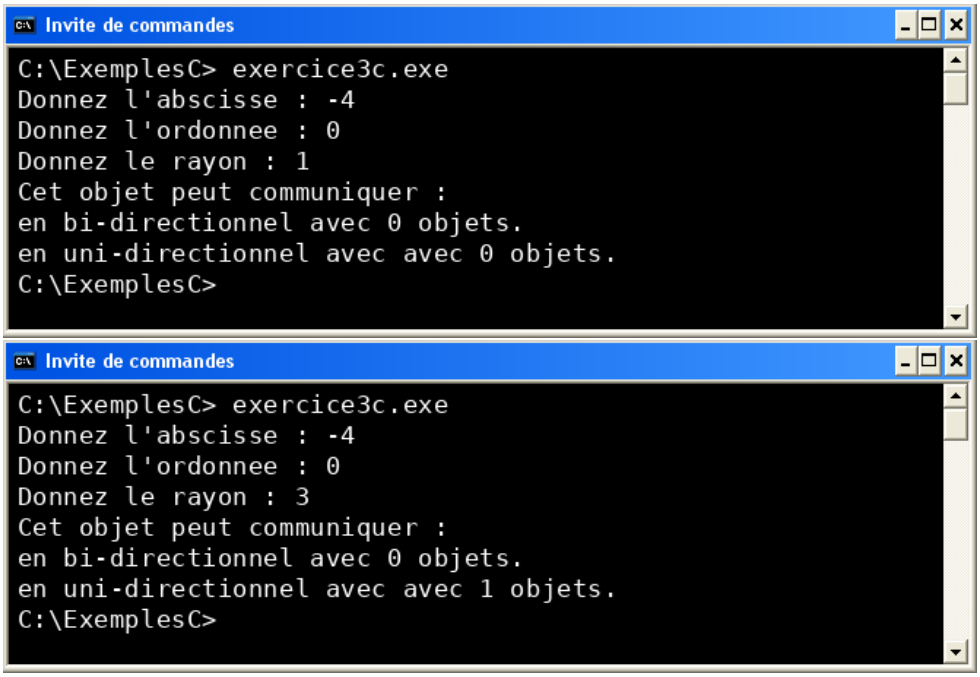
```
int bx[8] = { -2, 3, -4,  5, -5, 1,  0, 2 } ;
int by[8] = {  5, 5, -2, -2,  5, 3, -1, 2 } ;
int br[8] = {  2, 1,  1,  1,  4, 2,  2, 4 } ;
```

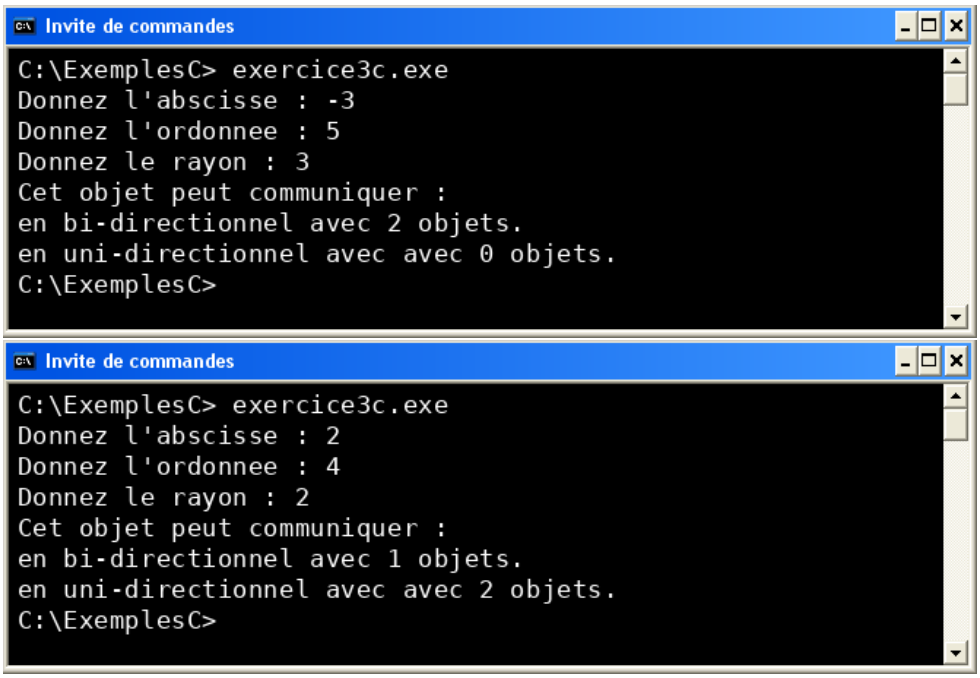
Par exemple, le premier objet communicant est positionné en (-2, 5) et le rayon de sa portée est de 2. Ces objets communicants et leurs portées ont été représentés graphiquement sur la figure suivante :



Le programme commencera par demander à l'utilisateur de saisir la position et le rayon de la portée d'un nouvel objet communicant, puis affichera (à l'aide d'une boucle) le nombre d'objets avec lesquels il peut communiquer de manière bidirectionnelle ou uni-directionnelle. Ce programme doit lui aussi utiliser la fonction du (a).

Ci-dessous, quatre exemples d'exécution de ce programme :





Écrivez ci-dessous le code du programme, puis vérifiez son exécution (une version déjà prête de la fonction sera ajoutée automatiquement lors du test, donc vous écrivez ici uniquement le programme demandé).