Exercice 1

Tracer et indenter chacun des quatre programmes suivants :

```
/* Programme 1*/

#include<stdio.h>

main(){
int n, B, c;

n=53058; //n comme nombre
B=10; //B comme base

//Point d'observation 1

while(n!=0){
c=n%B; //c comme chiffre
n=n/10; //Point d'observation 2
printf("%i", c);
}
printf("\n");
}
```

```
/* Programme 2*/

#include<stdio.h>

main(){
int n, B, c;

n=53058; //n comme nombre
B=10; //B comme base

//Point d'observation 1

while(n!=0)
c=n%B; //c comme chiffre
n=n/B; //Point d'observation 2
printf("%i", c);
printf("\n");
}
```

```
/* Programme 3*/

#include<stdio.h>

main(){
int n, B, c;

n=31; //n comme nombre
B=2; //B comme base

//Point d'observation 1

while(n!=0){
c=n%B; //c comme chiffre
n=n/B; //Point d'observation 2
printf("%i", c);
}
printf("\n");
}

/* Programme 4*/

#include<stdio.h>

main(){
int n, B, c;

n=31; //n comme nombre
B=2; //B comme base

//Point d'observation 1

while(n!=0)
c=n%B; //c comme chiffre
n=n/B; //Point d'observation 2
printf("%i", c);
printf("\n");
}
```


Corrigé

La difficulté est de bien voir la différence entre présence et absence d'accolades. Pour indenter, notez que Emacs le fait automatiquement. C'est une solution envisageable en TD, mais pas à l'examen. D'après le cours, tracer un programme, c'est "à chaque point d'observation donner la valeur de toutes les variables". Mais les étudiants sont autorisés à omettre les variables qui ne changent jamais de valeur (comme B ici).

1.

Pt obs	n	c
1	53 058	Indéterminé
2	5 305	8
2	530	5
2	53	0
2	5	3
2	0	5

Les étudiants les plus perspicaces doivent sentir qu'il y a là un algorithme intéressant de "conversion en base 10".

2.

Pt obs	n	c
1	53 058	Indéterminé

Le programme boucle : on n'atteint jamais le point d'observation 2. NB : pour l'arrêter : CTRL C.

3.

Pt obs	n	c
1	$31 = 11111_2$	Indéterminé
2	$31 = 1111_2$	1
2	$31 = 111_2$	1
2	$31 = 11_2$	1
2	$31 = 1_2$	1
2	$31 = 0$	1

Premier contact avec la base 2. Savent-ils ce que c'est ? Doivent-ils le savoir du lycée ? Occasion de tâter ce terrain.

4.

Boucle comme en 2.

Exercice 2

QCM

1. gcc est :
   a. Un identificateur du langage C
   b. Une commande qu'on tape dans la fenêtre de commande
   c. Un mot-clef du langage C
   d. Un opérateur du langage C

2. while est :
   a. Un identificateur du langage C
   b. Une commande qu'on tape dans la fenêtre de commande
   c. Un mot-clef du langage C
   d. Un opérateur du langage C

3. Dans le programme `main(){int a ; a=3%5 ;}`, % est :
   a. Un identificateur du langage C
   b. Une commande qu'on tape dans la fenêtre de commande

c. Un mot-clef du langage C
d. Un opérateur du langage C

4. Dans le programme `main(){int a ; a=3%5 ;}`, a est :
a. Un identificateur du langage C
b. Une commande qu'on tape dans la fenêtre de commande
c. Un mot-clef du langage C
d. Un opérateur du langage C

5. = permet en langage C de :
a. réaliser une affectation
b. tester une égalité
c. comparer deux identificateurs
d. convertir un int en float

6. En typographie le caractère & s'appelle :
a. La luette
b. L'espagnolette
c. L'esperluette
d. L'arpette

Corrigé

Rappel : en C, il 6 "unités lexicales de base", encore appelés lexèmes : les identificateurs, les mot-clefs, les séparateurs, les constantes, les constantes de type chaîne, les opérateurs.

1. b

2. c

3. d

4. a

5. a

6. c. Mais qui veut gagner des millions ?

Exercice 3

Écrire un programme permettant d'afficher les nombres de 1 à n, ainsi que leur carré, et leur cube (soit trois colonnes). L'entier n est au choix de l'utilisateur. Écrire ce programme trois fois : avec `while`, avec `do ... while` et avec `for`.

Exercice 4

Écrire un programme qui demande deux entiers a et b à l'utilisateur et indique en retour si b divise ou non a.

Exercice 5

Écrire un programme qui lit une note n sur 20, puis affiche la mention correspondante : ajourné si $n < 10$, passable si $10 \leq n < 12$, assez bien si $12 \leq n < 14$, bien si $14 \leq n < 16$, et très bien sinon.

Corrigé

Deux solutions : ou bien tester avec des if emboités, ou bien avec des formules booléennes. La première solution est plus efficace, surtout si on connait les cas les plus fréquents, à mettre en premier. La deuxième solution est plus proches des spécifications, et peut-être plus lisible (question de goût).

```
#include<stdio.h>

main(){
  float note;

  printf("Votre note svp : ");
  scanf("%f", &note);

  // Avec des if emboite's :
  if (note < 0) printf ("Pas de notes negatives\n"); else
    if (note<10) printf ("Ajourne\n"); else
      if (note<12) printf("passable\n"); else
        if (note < 14) printf("AB\n"); else
          if (note < 16) printf("B\n"); else
            if (note <= 20) printf ("TB\n"); else
              printf("Pas de notes au dessus de 20\n");

  printf("\n\n");

  // Avec des formules booleennes :
  if (note < 0) printf ("Pas de notes negatives\n");
  if (note >= 0 && note < 10) printf ("Ajourne\n");
  if (note >= 10 && note < 12) printf("passable\n");
  if (note >= 12 && note < 14) printf("AB\n");
  if (note >= 14 && note < 16) printf("B\n");
  if (note >= 16 && note <= 20) printf("TB\n");
  if (note > 20) printf("Pas de notes au dessus de 20\n");
}
```

Exercice 6

Écrire un programme qui demande 3 nombres à l'utilisateur, et les affiche en retour, mais triés par ordre croissant. Idem pour 4 nombres.

Corrigé

Multitude de solutions. On peut demander aux étudiants de dénombrer les résultats possibles dans le cas général : $n!$ facons d'ordonner n entiers, soit 6 pour ce qui nous concerne. Conséquence intéressante : comme chaque if nous permet de discerner deux cas, k usages de if nous mènent à 2^k possibilités. Donc, à moins d'utiliser des ruses mathématiques plus ou moins scabreuses, des case, ou je ne sais quoi, il faut au moins 3 if pour trier trois entiers ($2^2 < 3! \leq 2^3$). De même, il faut au moins 5 if pour trier quatre entiers ($2^4 < 4! \leq 2^5$). On doit pouvoir trier 5 entiers avec seulement 7 if ($2^6 < 6! \leq 2^7$), mais là ça devient de la combinatoire ...

Noter que sans affectations, chaque solution doit donner lieu à un printf "spécialisé", et donc, il semble qu'on soit obligé d'utiliser $n!$ if.

```
#include<stdio.h>

main(){
  int a, b, c, d;
  int aa, bb, cc, dd;
  int t;

  printf("Entrez un entier a : ");
  scanf("%i", &a);
  printf("Entrez un entier b : ");
  scanf("%i", &b);
  printf("Entrez un entier c : ");
  scanf("%i", &c);
  printf("Entrez un entier d : ");
  scanf("%i", &d);


  aa=a; bb=b; cc=c; dd=d; //On sauvegarde les valeurs pour faire plusieurs tests

  // Premiere methode : lourd
  if (a <= b && b <= c) printf("%i, %i, %i\n", a, b, c);
  if (a <= c && c <= b) printf("%i, %i, %i\n", a, c, b);
  if (b <= a && a <= c) printf("%i, %i, %i\n", b, a, c);
  if (b <= c && c <= a) printf("%i, %i, %i\n", b, c, a);
  if (c <= a && a <= b) printf("%i, %i, %i\n", c, a, b);
  if (c <= b && b <= a) printf("%i, %i, %i\n", c, b, a);


  a=aa; b=bb; c=cc; d=dd; //On remet à jour les valeurs

  //Une solution mixte
  if (a>b) {t=a; a=b; b=t;} //Maintenant, a<=b.
  if (c<=a) printf("%i, %i, %i\n", c, a, b); else
    if (c<=b) printf("%i, %i, %i\n", a, c, b); else
      printf("%i, %i, %i\n", a, b, c);


  a=aa; b=bb; c=cc; d=dd; //On remet à jour les valeurs
```

```
  //Une solution qui trie vraiment
  if (a>b) {t=a; a=b; b=t;} //Maintenant, a<=b. restent 3 cas
  if (a>c) {t=a; a=c; c=t;} //Maintenant, a<=c. restent 2 cas
  if (b>c) {t=b; b=c; c=t;} //Maintenant, b<=c.

  //Maintenant a<=b<=c
  printf("%i, %i, %i\n", a, b, c);


  a=aa; b=bb; c=cc; d=dd; //On remet à jour les valeurs

  //Une solution pour 4 nombres avec seulement 5 if
  if (a>b) {t=a; a=b; b=t;} //Maintenant, a<=b. restent 12 cas
  if (c>d) {t=c; c=d; d=t;} //Maintenant, c<=d. restent 6 cas
  if (b>d) {t=b; b=d; d=t;} //Maintenant, b<=d. restent 3 cas
  if (a>c) {t=a; a=c; c=t;} //Maintenant, a<=c. restent 2 cas
  if (b>c) {t=b; b=c; c=t;} //Maintenant, b<=c.

  //Maintenant a<=b<=c<=d
  printf("%i, %i, %i, %i \n", a, b, c, d);

  //Peut-on ecrire une solution aussi simple pour 5 nombres ? Mystere...
}
```

Exercice 7

On décrit le jeu des allumettes : au départ, il y a un tas de 50 allumettes, (ou tout autre objet : cailloux, jetons, ...). Chacun à son tour, les deux joueurs ôtent obligatoirement entre 1 et 6 allumettes. Celui qui ôte la dernière allumette gagne.

1. Préférez-vous commencer ou jouer en deuxième ? Justifiez votre choix par une stratégie gagnante. Conseil : commencer par raisonner avec un petit nombre d'allumettes, puis généraliser.

2. Écrire un programme qui joue au jeu des allumettes contre l'utilisateur.

Exercice 8

Exercice de révision : jusqu'où irez vous ? À chacune des questions suivantes, on demande d'écrire un programme. On conseille à chaque fois d'enregistrer, de compiler et d'exécuter le programme, de toujours travailler sur le même fichier et de faire "grossir" le programme de questions en questions.

Si vous voulez vous auto-évaluer : faites ce test en salle machine. Comptez deux points pour les questions 1 et 2, trois pour les suivantes. Ca vous fera une note assez approximative sur 19.

1. Écrire un programme qui affiche "Bonjour".

2. Faire en sorte que le programme demande à l'utilisateur d'entrer un nombre de son choix. Le programme doit ensuite afficher : "Le nombre que vous avez demandé est : *valeur*".

3. Le programme doit ensuite afficher un message indiquant si le nombre est oui ou non divisible par 7.

4. Le programme doit maintenant afficher la liste des diviseurs du nombre initialement entré. Par exemple, si l'utilisateur entre initialement le nombre 12, le programme doit écrire : "Les diviseurs de 12 sont 1, 2, 3, 4, 6 et 12".

5. On rappelle qu'un nombre est premier s'il n'est divisible que par 1 et par lui-même. Par convention, 1 n'est pas considéré comme un nombre premier. Le programme doit indiquer si le nombre entré par l'utilisateur est ou non premier.

6. Le programme doit maintenant afficher la liste de tous les nombres premiers inférieurs au nombre entré par l'utilisateur.

7. Le programme doit mainteant afficher seulement les nombre premier jumeaux. Deux nombres premiers sont dits *jumeaux* si leur différence est égale à 2 (ou à -2).

Remarque : à l'heure actuelle, personne ne sait s'il existe un nombre fini ou infini de nombres premiers jumeaux.