1. * Crible d'Ératosthène

Partie 1

a. Affiche un tableau 10 x 10 contenant les 100 premiers nombres entiers.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

b. Au premier clic raye les multiples de 2 sauf 2.

c. Au clic suivant raye les multiples de 5 sauf 5.

d. Au clic suivant raye les multiples de 3 sauf 3.

Explique pourquoi il n'est pas nécessaire de rayer les multiples de 4.

Cite trois nombres dont tous les multiples sont déjà rayés.

e. Au clic suivant raye les multiples de 7 sauf 7.

Partie 2

Le but est d'écrire un programme qui détermine tous les nombres premiers inférieurs à N.

On testera si un nombre X est divisible par un des nombres premiers déjà connus.

Doit-on tester tous les nombres ou peut-on en éviter ?

2. Cryptographie de César

On appelle cryptographie les méthodes pour coder les messages. Les premiers écrits stipulant l'utilisation de cryptographie datent de l'époque romaine avec l'apparition du « chiffrement de César ». On l'utilisait pour donner des ordres aux troupes afin que les ennemis ne puissent pas déchiffrer les messages et anticiper les mouvements des armées.
Un message crypté par cette méthode débutait par un nombre puis le message dont les lettres avaient été décalées de ce nombre de rang dans l'ordre alphabétique.
Par exemple, le message : « 3 M'DLPH OHV PDWKV » signifie « J'AIME LES MATHS »
(le 3 signifie que l'on a décalé de trois rangs et donc un A est transformé en D, un B en E, etc).

Pour la programmation, on utilise deux fonctions :

- Lettre-nombre() qui, quand on lui donne une lettre, rend le numéro de cette lettre dans l'alphabet.
- Nombre-lettre() qui, quand on lui donne un nombre entre 1 et 26, rend la lettre de l'alphabet correspondante.

Voici des extraits de ces deux fonctions :

```
Lettre-nombre(lettre)
    if lettre=='a' : return(1)
    if lettre=='b' : return(2)
,,,,,,,,
```

```
Nombre-lettre(nombre)
    if nombre==1 :
return(a)
    if nombre==2 :
return(b)
,,,,,,,,
```

1 Crée les deux fonctions Lettre_nombre et Nombre-lettre,

2 Crée un programme qui crypte un message avec un décalage de 3 lettres.

3 Crée un algorithme qui, quand on lui donne un texte et un nombre, crée un message crypté par le chiffrement de César.

4 Imagine une autre façon de remplacer lettre par lettre par simple décalage et crée un algorithme correspondant.

3. Bataille navale

Voici une grille de jeu :

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										
K										
L										

Navires :

- 1 porte-avions (5 cases)
- 1 croiseur (4 cases)
- 1 contre-torpilleurs (3 cases)
- 1 sous-marin (3 cases)
- 1 torpilleur (2 cases)

1 Écris un programme qui place les bateaux de bataille navale dans un tableau de façon aléatoire.

2 Écris un programme qui te permet de jouer contre l'ordinateur à la bataille navale (tu tires). L'ordinateur dispose d'une grille fixée (comme au **1**) et te répond « touché, coulé ou dans l'eau ».

3 Écris un programme qui te permet de jouer à la bataille navale (l'ordinateur tire). Tu disposes d'une grille de jeu. L'ordinateur te propose des cases et tu réponds T, C ou O pour « touché, coulé ou dans l'eau ».

4 Écris un programme qui reprend les trois précédents et te permet de jouer contre l'ordinateur à la bataille navale (tu tires, puis c'est l'ordinateur, chacun son tour).

4. Calendriers pour des années allant de 1900 à 2048

1 Écris un programme qui donne le jour de la semaine pour la saisie d'une date (le format étant :jj/mm/aaaa).

2 * Écris un programme qui affiche le calendrier pour un mois donné saisi (le format étant :mm/aaaa). On peut utiliser l'exercice précédent.

3 Écris un programme qui affiche le nombre de jours écoulés entre deux dates.

5. Labyrinthes

1 Écris un programme qui affiche un labyrinthe (cases ou images) et dans lequel tu déplaces un disque à l'aide des quatre flèches du clavier. Lorsque tu touches les murs ou les bords, tu perds et reviens au départ.

2 Modifie le programme précédent : tu as au départ 5 points et perds 1 point à chaque erreur. Lorsque tu n'as plus de points, tu perds et reviens au départ.

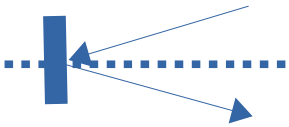
3 Modifie le programme précédent : tu comptes le nombre de déplacements effectués et affiches le score obtenu. Le meilleur score est sauvegardé. Le but est de sortir du labyrinthe avec le score minimal.

6. Collisions

*** Sur un mur**

1 Écris un programme qui déplace une balle suivant une trajectoire rectiligne aléatoire dans un cadre rectangulaire. La balle doit s'arrêter sur un bord.

2 Modifie ce programme pour que la balle continue après rebond sur un des bords. Le rebond se fait par symétrie par rapport à la perpendiculaire au bord, au point de contact.



3 Modifie ce programme pour faire afficher le rebond de la balle sur tous les bords du cadre.

Sur un objet

1 Écris un programme qui déplace une balle à la souris (ou au clavier) en évitant la position fixe d'un carré.

2 Modifie ce programme pour faire éviter un disque fixe.

3 Modifie ce programme pour faire éviter deux obstacles : un carré et un disque.

Pong

1 Écris un programme de « pong » à un joueur contre un mur.

2 Écris un programme de « pong » à deux joueurs.

7. Entraîne-mémoire

Un damier contenant des objets est montré pendant un temps fixe.
Le damier est ensuite vidé et le programme propose les objets un par un.
Le joueur doit cliquer sur la case contenant cet objet, il marque alors un point.
Le but est de trouver tout le damier (16 points : « gagné »).
Pour chacune des parties suivantes écris un programme.

Partie 1 Damier 4X4 avec des nombres.

Le damier contient 16 nombres aléatoires.
Si la case est trouvée, elle reste affichée.
Si c'est une erreur, on affiche la case pendant un temps bref.

Partie 2 Avec des mots.

Une liste de 16 mots est utilisée. Sa composition permet de faire évoluer la difficulté du jeu.

Partie 3 Avec des images

Une liste de 16 images est utilisée. Sa composition permet de faire évoluer la difficulté du jeu.

Partie 4 Niveau de difficulté

Pour rendre le jeu plus compliqué :

- Si la case est trouvée elle n'est pas affichée.
- Si c'est une erreur, on n'affiche rien.
- On augmente le nombre de cases.
- On minute le temps de réponse ...

8. Statistiques

On utilisera une série statistique numérique donnée par la liste de valeurs du caractère et la liste des effectifs.

Partie 1

Les listes de départ sont données.

1 Calcule « Unite » : le plus petit écart entre deux modalités.

2 Établis la liste des fréquences.

3 Calcule et affiche la valeur moyenne de la série.

Partie 2

Écris un programme de saisie des deux listes de départ.
Reprends les questions de la partie 1.

Partie 3

Établis la liste des fréquences cumulées croissantes.

Détermine et affiche la valeur médiane de la série.

Partie 4

1 * Affiche un diagramme à barres des effectifs.(Utilise le 1 de la partie 1).

2 * Affiche un diagramme circulaire des effectifs.

3 * Reprends 1 et 2 avec les fréquences.

9. Trésor caché

Un trésor est positionné de façon aléatoire dans un cadre (écran) et non visible.

Partie 1

Le joueur clique sur une position.
Le programme répond :
« gagné » si c'est sur le trésor.
sinon il répond :
« plus à gauche » ou « plus à droite »
« plus haut » ou « plus bas ».

Écris ce programme.

Partie 2

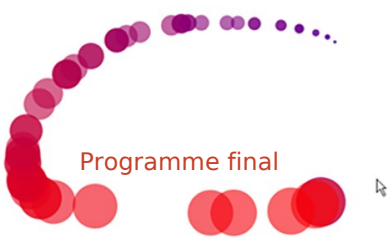
Le joueur clique sur une position.
Le programme répond :
« gagné » si c'est sur le trésor
sinon il donne la distance (en pixels) entre le point cliqué et le trésor.

Écris ce programme.

10. * Animations

Trace Ball

1 Écris un programme qui anime un disque à la souris et garde la trace des 60 dernières positions.



2 Modifie ce programme pour faire afficher aussi les dernières positions avec des disques de plus en plus petits.

3 Modifie ce programme pour faire afficher les dernières positions avec des disques de couleurs proches mais différentes.