


```

void affiche_rectangle(int n){
    for(int i=0;i<n;i+=1){
        printf("*");
    }
    printf("\n");

    for(int i=1;i<n-1;i+=1){
        printf("*");
        for(int j=1;j<n-1;j+=1){
            printf(" ");
        }
        printf("*\n");
    }

    for(int i=0;i<n;i+=1){
        printf("*");
    }
    printf("\n");
}

```

```

void affiche_losange(int n){
    for(int j=0;j<n;j+=1){
        for(int i = 0;i<2*n-1;i+=1){
            if( (i<=n-2-j) || (i>=n+j)){
                printf(" ");
            }
            else{
                printf("*");
            }
        }
        printf("\n");
    }

    for(int j=n-2;j>=0;j-=1){
        for(int i = 0;i<2*n-1;i+=1){
            if( (i<=n-2-j) || (i>=n+j)){
                printf(" ");
            }
            else{
                printf("*");
            }
        }
        printf("\n");
    }
}

```

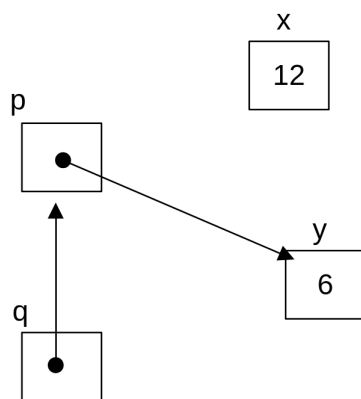
7. Qu'est-ce qu'un pointeur? Dessiner l'état de la mémoire (pas besoin de faire apparaître les adresses) après les instructions suivantes :

Un pointeur est une variable dont la valeur est une adresse mémoire. On dit qu'il pointe vers la case à cette adresse.

```

void f(int x){
    int* p = &x;
    *p = 5;
    int** q = &p;
    int y = 4;
    *q = &y;
    y=6;
    x=12;
}

```



Exercice 2 Un peu de bash

On suppose que l'on se trouve à la racine du répertoire personnel `/home/bob` de Bob. On ne se déplacera pas dans un autre répertoire sauf si cela est explicitement demandé.

Vous pouvez utiliser la commande suivante :

- `cp chemin1 chemin2` copie le dossier ou fichier correspondant au chemin `chemin1` à l'endroit indiqué par le chemin `chemin2`. Exemple : si dans le répertoire courant il existe un fichier `original.txt`, alors la commande `cp original.txt /dossier_exemple/copie.txt` copie le fichier dans le dossier `dossier_exemple` situé dans le répertoire personnel et le renomme `copie.txt`.

Traduire chacune des requêtes suivantes de manière à pouvoir les exécuter successivement sur un terminal en bash :

1. Lister le contenu du répertoire courant. **ls**
2. Lister toutes les informations sur le fichier **fichier_secret_bob.txt** qui est dans le répertoire courant. **ls -l fichier_secret_bob.txt**

On suppose que le résultat est le suivant :

```
-r-rw-rw- 1 bob amis_bob 15466 1 janv. 2024 fichier_secret_bob.txt
```

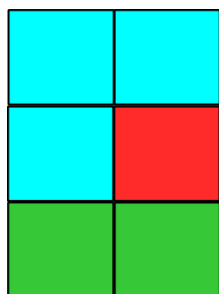
3. Qui est le propriétaire du fichier? Bob
4. Que peuvent faire les amis de Bob avec le fichier? Lire et écrire
5. Changer les droits de ce fichier pour donner les droits en lecture et écriture à son propriétaire, les droits en lecture à son groupe et rien aux autres. **chmod u+w,g-w,o-rw**
6. Créer un répertoire vide de nom **info** dans le répertoire courant. **mkdir info**
7. Aller dans ce répertoire. **cd info**
8. Créer un fichier vide **colle.txt**. **touch colle.txt**
9. Retourner dans le répertoire personnel de Bob. **cd ..**
10. Créer un répertoire vide **physique** dans le répertoire courant. **mkdir physique**
11. Copier le fichier **colle.txt** dans ce répertoire. **cp info/original.txt physique/colle2.txt**
12. Aller dans **info**. **cd info**
13. Sans changer de répertoire, donner les droits en lecture seule à tout le monde pour le répertoire **physique**. **chmod a=r ../physique**

Exercice 3 Un peu de binaire

1. Comment écrit-on 247 en binaire? Et 1459? 11110111 et 10110110011
2. Qu'est ce qu'un octet? Combien d'octets faut il pour stocker le nombre 123456789?
Un octet est un paquet de 8 bits. Il faut $\lfloor \log_2(123456789) \rfloor + 1 = 27$ bits pour stocker ce nombre, donc 4 octets
3. Faire les opérations suivantes en binaire, en se limitant à un octet, et en expliquant la méthode et le résultat :
12+34, 124-56, 38-102
On trouve 101110 pour l'addition, il suffit de la poser.
Pour les soustractions, on ajoute le complément à 2 de 56 (et 102) à 124 (et 38), en ignorant les retenues supplémentaires.
On obtient les additions $01111100 + 11001000 = 1000100$ (124-56=68) et $00100110 + 10011010 = 11000000$ (38-102=-64, c'est à dire =64 en complément à 2)

Informatiquement parlant, une image est une matrice de pixels. Le format d'image bitmap **.bmp** se code avec 3 octets par pixel, représentant chacun le niveau de rouge, de vert et de bleu dans la couleur du pixel. Il y a une subtilité supplémentaire, qu'on ignorera.

Par exemple l'image suivante de taille 2*3 devrait être représentée par (les valeurs RGB sont en décimal pour que ça soit plus lisible) :



0 255 255	0 255 255
0 255 255	255 42 42
55 200 55	55 200 55

4. Combien d'octets sont nécessaires pour une image de taille 2500*2500? Quelle est la taille maximale (en nombre de pixels) d'une image pouvant être stockée sur 2.0 Go, sur 250 Mio?
Une image 2500*2500 signifie $2500 * 2500 * 3 = 18750000$ octets. On note n le nombre de pixels d'un image, il faut $n * 3$ octets pour la stocker. On cherche donc le plus grand n tel que $3 * n \leq 2 * 10^9$, soit $n = 666666666$ et le plus grand n tel que $3 * n \leq 250 * 2^{20}$, soit $n = \lfloor 250 * 2^{20} / 3 \rfloor = 87381333$.

Exercice 4 Lecture de code

On considère les codes suivants

1. Faire un tableau avec les valeurs de x , y et i avant la boucle, à chaque tour de boucle et après la boucle, pour les entrées $x = 12$, $y = 937500$.

```

int f(int x, int y){
    int i=0;
    while(x>=0){
        y=y/5;
        i+=1;
        x=x-i;
    }

    if(i>6){
        y=y/2;
    }
    else{
        y=y-1;
    }
    return y
}

```

	i	x	y
Avant la boucle	0	12	937500
	1	11	187500
	2	9	37500
	3	6	7500
	4	2	1500
	5	-3	300
À la fin	5	-3	299

2. Donner l’affichage complet (avec les retours à la ligne) de cette fonction pour $n = 5$.
Que fait cette fonction ?

```

void mystere(int n){
    int i = 0;
    int a = 1;
    int j, b, c;

    while (i < n){
        j = 0;
        b = 1;
        c = a;

        while (j < i){
            printf("%d ", a/(b*c));
            c = c/(i-j);
            j = j + 1;
            b = b*j;
        }

        printf("%d ", a/(b*c));
        printf("\n");
        i = i + 1;
        a = a*i;
    }
}

```

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

```

Elle affiche le triangle de Pascal.

Exercice 5 Un peu de récursivité

On définit une fonction A sur $\mathbb{N} \times \mathbb{N}$ par :

$$\begin{cases} A(0, p) = p + 1 & \text{pour } p \geq 0 \\ A(n, 0) = A(n - 1, 1) & \text{pour } n \geq 1 \\ A(n, p) = A(n - 1, A(n, p - 1)) & \text{si } n \geq 1, p \geq 1 \end{cases}$$

1. Calculer $A(0, 0)$ puis $A(2, 2)$. Il est conseillé d'être astucieux.

$A(0, 0) = 1$ par définition.

Pour le calcul de $A(2, 2)$, on va commencer par montrer que $A(1, p) = 2 + p$ par récurrence sur p .

Pour $p = 0$, on a $A(1, 0) = A(0, 1) = 2$.

Supposons $A(1, p) = 2 + p$ pour un certain p . Alors $A(1, p + 1) = A(0, A(1, p)) = A(1, p) + 1 = 2 + p + 1$. D'où l'hérédité.

$$\begin{aligned} A(2, 2) &= A(1, A(2, 1)) = A(1, A(2, 0)) + 2 \\ &= A(2, 0) + 4 = A(1, 1) + 4 \\ &= 7 \end{aligned}$$

2. Implémenter la fonction A en C. La signature sera `fct_A(int n, int p)`.

On vérifiera que les entrées n et p sont valides avec l'instruction `assert`.

```
int Ackermann(int n, int p){
    assert(n>=0);
    assert(p>=0);
    if(n==0){return p+1;}
    else if (p==0){return Ackermann(n-1,1);}
    else {return Ackermann(n-1,Ackermann(n,p-1));}
}
```

Remarque : la fonction A (qui s'appelle en fait "fonction d'Ackermann") croît très rapidement. Par exemple, $A(4, 2) > 10^{10000}$.

Exercice 6 Un peu de morpion

Dans cet exercice on va écrire un petit programme C qui permet à deux joueurs de jouer au morpion au clavier.

On utilisera pas un tableau, mais plutôt 9 variables représentant les cases du plateau : $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$. Elles peuvent prendre trois valeurs : le caractère espace ' ' si la case est vide, 'o' si la case contient un rond et 'x' si la case contient une croix.

Par exemple :

X		X
	o	o
		o

est représenté par $x_0 = 'x', x_1 = ' ', x_2 = 'x', x_3 = ' ', x_4 = 'o', x_5 = 'o', x_6 = ' ', x_7 = ' ', x_8 = 'o'$.

1. Écrire une fonction `void affiche_grille(char x0, char x1, char x2, char x3, char x4, char x5, char x6, char x7, char x8)`, qui affiche la grille ainsi (toujours sur le même exemple) :

X		X
-	-	-
	o	o
-	-	-
		o

```
void affiche_grille(char x0, char x1, char x2, char x3, char x4, char x5, char x6, char x7, char x8){
    printf("%c| %c| %c", x0, x1, x2);
    printf("- - -");
    printf("%c| %c| %c", x3, x4, x5);
    printf("- - -");
    printf("%c| %c| %c", x6, x7, x8);
}
```

2. Écrire une fonction

`bool est_gagnante(char x0, char x1, char x2, char x3, char x4, char x5, char x6, char x7, char x8)`

qui teste s'il y a un alignement de 3 ronds ou de 3 croix. La fonction renvoie `true` et affiche qui a gagné si c'est le cas. Sinon la fonction renvoie `false`.

```

bool est_gagnante(char x0, char x1, char x2, char x3, char x4, char x5, char x6, char x7, char x8){
    // Conditions sur les lignes
    if (x0==x1 && x1==x2 && x0!=' '){printf("%c a gagné",x0); return true;}
    if (x3==x4 && x4==x5 && x3!=' '){printf("%c a gagné",x3); return true;}
    if (x6==x7 && x7==x8 && x6!=' '){printf("%c a gagné",x6); return true;}

    //Conditions sur les colonnes
    if (x0==x3 && x3==x6 && x0!=' '){printf("%c a gagné",x0); return true;}
    if (x1==x4 && x4==x7 && x1!=' '){printf("%c a gagné",x1); return true;}
    if (x2==x5 && x5==x8 && x2!=' '){printf("%c a gagné",x2); return true;}

    //Conditions sur les diagonales
    if (x0==x4 && x4==x8 && x0!=' '){printf("%c a gagné",x0); return true;}
    if (x2==x4 && x4==x6 && x2!=' '){printf("%c a gagné",x2); return true;}

    //Pas d'alignement
    return false;
}

```

On propose la fonction suivante qui permet de demander à un utilisateur de rentrer le numéro d'une case et qui la renvoie :

```

int entre_coup(){
    int x;
    scanf("%d\n", &x);
    assert (x<=8);
    assert (x>=0);
    return x;
}

```

```

3. int main(){
    int x0 = ' ';
    //On suppose que cette même ligne a été recopiée pour x1,...,x8.

    bool fini = false;
    nb_coups = 0;
    joueur = 'o';
    affiche_grille(x0,...,x8); //j'élude les nombreuses entrées
    while (!fini){

        //jouer le coup
        printf("Entrez un coup\n");
        int coup = entre_coup();
        if(coup == 0 && x0 == ' '){x0=joueur;}
        else if(coup==1 && x1 == ' '){x1=joueur;}
        //On suppose la même ligne recopiée pour x2, ...,x8
        else {printf("Coup invalide, erreur\n"); break;} //Optionnel

        //Affichage
        affiche_grille(x0,...,x8); //j'élude les nombreuses entrées

        //Vérifier si le jeu est fini
        if (est_gagnante(x0,...,x8)){ //j'élude les nombreuses entrées
            fini = true;
        }
        //Une autre maniere de finir le jeu est si on a joué 9 coups et la partie est nulle
        nb_coups +=1;
        if (nb_coups == 9){
            fini = true;
        }

        //Changer de joueur pour le tour suivant
        if(joueur=='o'){
            joueur='x';
        }
        else {joueur = 'o';}
    }
}

```