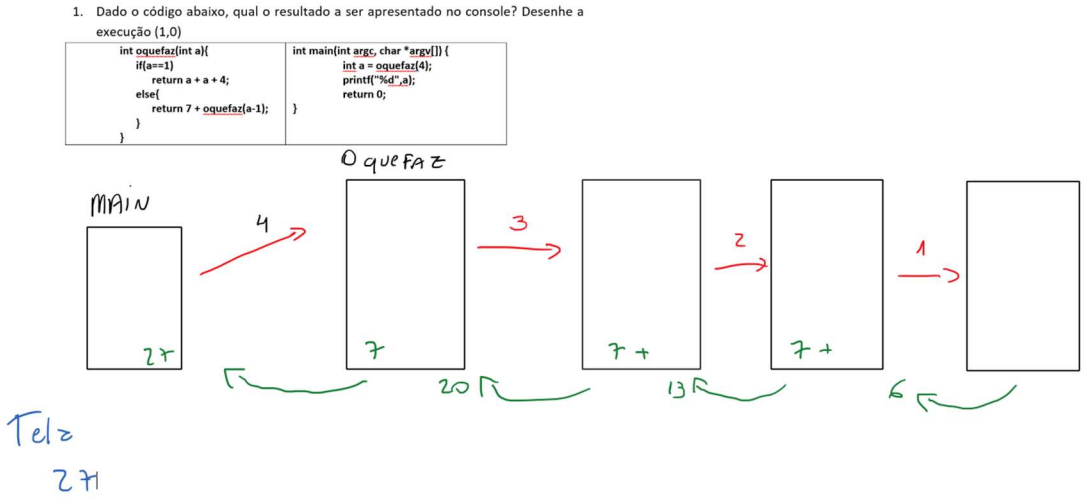
Prova 2

1. Dado o código abaixo, qual o resultado a ser apresentado no console? Desenhe a execução (1,0)

int oquefaz(int a){ **if(a==1)** **return a + a + 4;** **else{** **return 7 + oquefaz(a-1);** **}** **}**	**int main(int argc, char *argv[]) {** **int a = oquefaz(4);** **printf("%d",a);** **return 0;** **}**



2. Faça um algoritmo que peça um número inteiro e calcule a quarta potência, usando recursividade. Use a estrutura básica de uma função recursiva que realiza cálculos e retorna um valor ensinada em aula (1,5).

```
#include<stdio.h>
#include<stdlib.h>
#include<locale.h>

int calcular(int a, int contador){
            if(contador==3){
    return a;
} else{
    a *= calcular(a,contador+1);
}

    return a;
}

int main (){
```

```
        setlocale(LC_ALL, "Portuguese");
        int a;

        printf("Cálculo da quarta potencial de um número inteiro\n ");
        printf(" Inserir número: ");
        scanf("%d" ,&a);
        a = calcular(a,0);
        printf("\n%d" ,a);

    }
```

3. Transforme o código a seguir em uma função recursiva (1,5):

```
void fazAlgo(){
   for(int i=1;i<4;i++){
      if(i%2==0)
         printf("\n%d é par",i);
      else
         printf("\n%d é ímpar",i);
   }
}
```

```
#include<stdio.h>
#include<stdlib.h>
#include<locale.h>
void fazAlgo(int a){

    if (a==3){
    printf("\n%d é ímpar",a);

}
    else{
        fazAlgo(a+1);
        if(a%2==0){
            printf("\n%d é par",a);
        }
        else{
            printf("\n%d é ímpar",a);
        }
  }

}
```

```
int main (){
    setlocale(LC_ALL, "Portuguese");
    fazAlgo(1);

    return 0;
}
```

4. Transforme o código a seguir em uma função não recursiva (1,5):

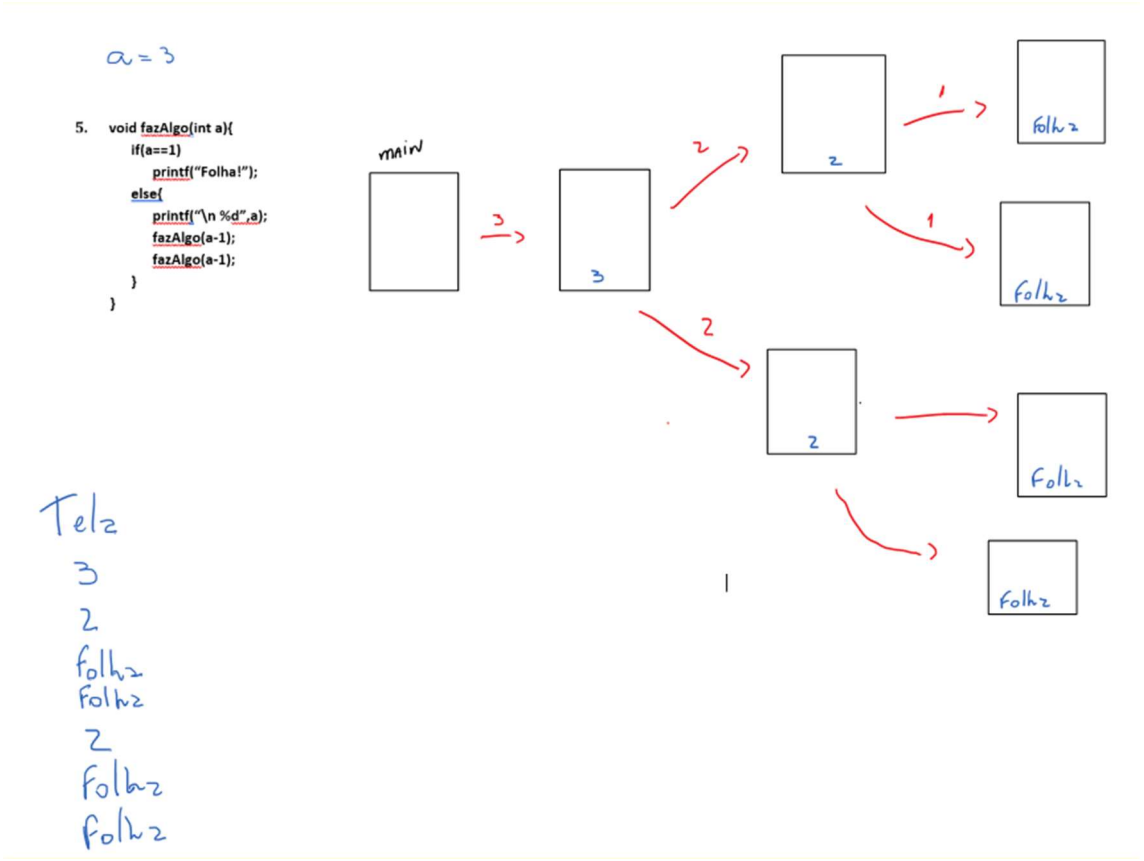
```
void fazAlgo(){
    int a;
    printf("\nDigite algo:");
    scanf("%d",&a);
    switch(a){
      case 1:
        printf("\nVocê digitou 1");
        break;
      case 2:
        printf("\nVocê digitou 2");
        break;
      case 0:
        printf("\nVocê digitou 0. Tchau!");
        break;
    }
    if(a!=0){
      fazAlgo();
    }
}
```

```
#include<stdio.h>

#include<stdlib.h>

#include<locale.h>


void fazAlgo(){

    int a;

    do{

        printf("\nDigite algo:");

        scanf("%d",&a);


    switch(a){
```

```
        case 1:
                printf("\nVocê digitou 1");
                break;
        case 2:
                printf("\nVocê digitou 2");
                break;
        case 0:
                printf("\nVocê digitou 0. Tchau!");
                break;
        }

        }while (a!=0);
}
int main(){
        fazAlgo();
        return 0;
}
```

5.
```
void fazAlgo(int a){
    if(a==1)
        printf("Folha!");
    else{
        printf("\n %d",a);
        fazAlgo(a-1);
        fazAlgo(a-1);
    }
}
```

Supondo que a função *main()* chame fazAlgo*(3)*, desenhe a execução. (1,5)



6. **Considere o trecho de código abaixo e assinale V para verdadeiro ou F para falso(1,2).**

1. int *a;
2. a = (int*) malloc(3*sizeof(int));
3. a[0] = 3;
4. *(a+1)=4;
5. a[2] = 5;
6. printf("%d %d %d",a[0],a[1],*(a+2));

(v) A linha 1 está correta; (v) A linha 2 está correta; (v) A linha 3 está correta;	(v) A linha 4 está correta; (v) A linha 5 está correta; (v) A linha 6 está correta;

5- Considere o código abaixo:

```
void inicializar(int **a){
        a = (int*)malloc(sizeof(int));
}
int main(int argc, char** argv) {
        int *b;
        inicializar(b);
         *b=10;
        return 0;
}
```

7. O código está correto (0,3)? Justifique (0,5).

R: Não está certo, pois:

Primeiro: Para inicializar um ponteiro com alocação dinâmica em uma função, é necessário passar o endereço do ponteiro da main: Passagem por referência.

Segundo: Na função inicializar, o ponteiro de ponteiro, ‘a’, receberia o endereço do ponteiro ‘b’, e para inicializar o ponteiro ‘b’, o ponteiro ‘a’ deveria estar utilizando o espaço de b, mas não é o caso. O ponteiro ‘a’ está utilizando o espaço dele mesmo, que é focado em receber um ponteiro de ponteiro, para colocar a alocação dinâmica, e isso é errado. Deveria ser *a = (int*)malloc(sizeof(int));