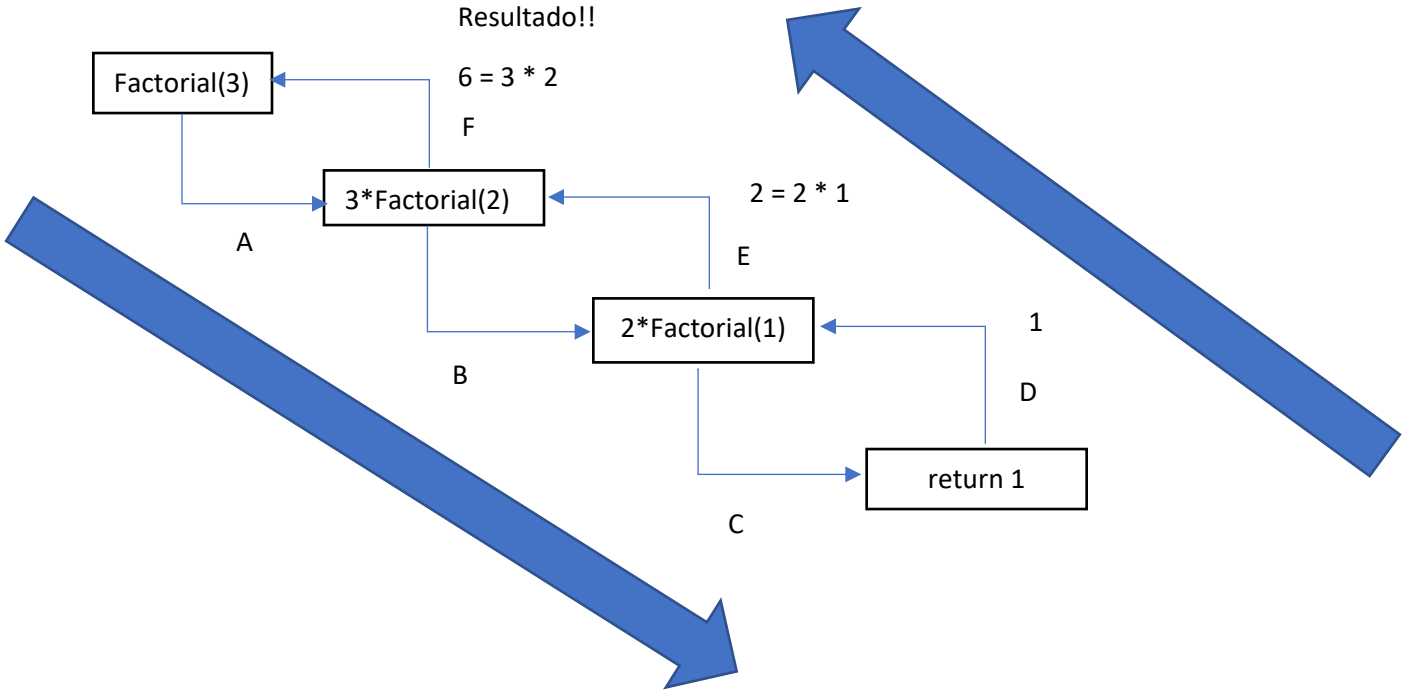
Además responder:

Sea el clásico algoritmo recursivo factorial(n)

```
int factorial(int n)
{
    if (n==0 || n==1)
       return 1;
    else
       return(n*factorial(n--1));
}
int main(int argc, char* argv[])
{    int N = 3;
    printf("Factorial: %i %i\n", N, factorial(N));
    return 0;}
```

Desarrolle :

1. ***Paso a paso(a pie) la corrida de un algoritmo recursivo Factorial(3) … usando el diagrama de llamados como hace los llamados recursivos y los retornos***



2. ***Indique en el código fuente anterior el punto de parada y punto de continuidad(Señale y explique su posición)***

La condición de parada sería: if (n==0 || n==1) {return 1}; Esta indicaría el fin del ciclo ya que cuando n (número digitado por el usuario) sea igual a 0 o a 1, entonces devolvería un 1 y se dejaría de llamar de manera continua a la misma función, rompiendo así el ciclo y devolviendo los valores obtenidos multiplicados por 1.

El punto de continuidad sería: else{return(n*factorial(n--1))}; Esta sería la condición que continua el ciclo devolviendo valores de manera constante hasta cumplir con la condición de parada, si esta no se cumple.

3. ***Que pasa si no se especifica bien el caso límite o caso base??***

En caso de no especificarse el limite de parada, entonces el ciclo seguirá ejecutándose sin ningún fin, por lo que el programa no pararía jamás, es decir, no termina.

4. ***Defina que es una Subrutina recursiva. Por qué se dice que programar recursivamente es programar en 3D?***

Una subrutina recursiva es una que se llama constantemente a sí misma hasta encontrar una condición de parada.

Es 3D porque con cada llamado se va haciendo profunda esa función recursiva, llegando hasta caso límite y devolviendo todos los valores que ha acumulado durante los llamados que ha tenido.

5. ***Enumere ventajas y desventajas de programar recursivamente.***

- Desventajas:
  a. Los algoritmos son algo difíciles de plantear y mantener debido a lo abstracto que es, por lo que no es posible programar cualquier cosa de manera recursiva.
  b. Se debe tener un planteamiento claro acerca del algoritmo y su caso límite ya que si no se realiza correctamente el programa nunca terminará y consumirá gran parte de la memoria (sino toda) debido a la réplica de variables locales, parámetros e información de control de la pila de llamados.
- Ventajas:
  a. Ideal para algunos comportamientos o algoritmos computacionales.
  b. Normalmente lo que se programa en recursividad se puede hacer con un ciclo.

6. ***Se puede decir que todo lo que se programa recursivamente se puede programar con ciclos… pero al revés también?***

Al revés no porque se comporta de manera iterativa o repetitiva, implicando que en cada llamado se guarden todos lo valores, variables, espacios de memoria ocupados para los parámetros y las variables locales. Hace que cada llamada recursiva deje pausado su estado para poder invocar de nuevo una función de igual nombre, volviendo a llamarse a ella misma llegando a consumir mucha memoria, debido a eso no todo ciclo se puede programar recursivamente.