1. A dica aqui é observar que o padrão está desenhado na horizontal, mas a funções substitutas podem mostrar o padrão na vertical!

```
void pontoRolo1 ()
{
    printf ("v");
}

void pontoRolo2 ()
{
    printf ("a");
}

void moveAgulha ()
{
    printf (" ");
}

void rolaTecido ()
{
    printf ("\n");
}
```

2.

```
#define LARGURA_FAIXA 6

int main ()
{
    int i, n_pontos = 0;

    while (1)
    {

        for (i = 0; i < LARGURA_FAIXA; i++)
        {
            if (i < n_pontos)
                pontoRolo1 ();
            else
                pontoRolo2 ();
        }

        if (n_pontos >= LARGURA_FAIXA)
            n_pontos = 0;
        else
            n_pontos++;

        rolaTecido ();
    }

    return (0);
}
```

3.

```
#define LARGURA_FAIXA 6

void main ()
{
    int n_pontos = 1, usa_rolo1 = 0;

    while (1)
    {
        int i;

        for (i = 0; i < n_pontos; i++)
        {
            if (usa_rolo1)
                pontoRolo1 ();
            else
                pontoRolo2 ();
        }

        n_pontos++;
        if (n_pontos > LARGURA_FAIXA)
        {
            n_pontos = 1;
            usa_rolo1 = !usa_rolo1;
        }

        rolaTecido ();
    }
}
```

4.

Tabela:

	0	1	2	3	4	5	6	7	8	9	10	11
p1	0	1	2	3	4	5	4	3	2	1	0	1
p2	5	4	3	2	1	0	1	2	3	4	5	4
c1	1	1	1	1	1	-1	-1	-1	-1	-1	1	1
c2	-1	-1	-1	-1	-1	1	1	1	1	1	-1	-1

Padrão:

```
|v   a   v   a   v   a   v   a   v   a   v   a   v   a   v   a   v |(...)
| v  a a  v v  a a  v v  a a  v v  a a  v v  a a  v v  a a  v v  a a  v v|(...)
|  va   av   va   av   va   av   va   av   va   av   va   av   va   av  |(...)
|  av   va   av   va   av   va   av   va   av   va   av   va   av   va  |(...)
| a  v v  a a  v v  a a  v v  a a  v v  a a  v v  a a  v v  a a  v v  a a|(...)
|a   v   a   v   a   v   a   v   a   v   a   v   a   v   a   v   a |(...)
```

5.

```
/* Usamos aqui a ideia de uma “janela deslizante”, com 3 valores. */

int main ()
{
    int n1, n2, n3, i;

    /* Sempre trabalhamos com 3 números. */
    n1 = leProximoDado ();
    n2 = leProximoDado ();
    n3 = leProximoDado ();
    i = 2; /* O primeiro valor que verificamos é o 2o. */

    while (n3 >= 0)
    {
        if (n2 > n1 && n2 > n3) /* pico. */
            registraPadrao (i, PICO);
        else if (n2 < n1 && n2 < n3) /* vale. */
            registraPadrao (i, VALE);
        else if (n2 > n1 && n2 < n3) /* subida */
            registraPadrao (i, SUBIDA);
        else if (n2 < n1 && n2 > n3) /* descida */
            registraPadrao (i, DESCIDA);

        /* "Desliza" os valores. */
        n1 = n2;
        n2 = n3;
        n3 = leProximoDado ();
        i++;
    }

    return (0);
}
```

6.

```
/* Este problema parece muito complicado, mas fica muito mais simples se você
 observar que a função verificaFinal, que já criamos, resolve boa parte do
 problema. Para isso, precisamos primeiro ajustar os números de forma que a seja
 sempre maior que b (e lembrar disso, porque o retorno muda se trocarmos os
 valores). Depois disso, basta chamar a função verificaFinal para testar se b é
 o final de a. Se não for, descartamos o último dígito de a e repetimos o
 procedimento, até encontrarmos b no final de ou até a ficar menor ou igual a b,
 o que acontecer primeiro. */

#define ASEGB 1 // A segmento de B.
#define BSEGA 2 // B segmento de A.

int verificaSegmento (unsigned int a, unsigned int b)
{
    int trocou = 0; // Flag que diz se a e b trocaram de posição.
    unsigned int aux;

    // Caso especial: um precisa ser maior que o outro.
    if (a == b)
        return (0);

    if (b > a)
    {
        // Troca a e b para garantir que a > b.
        aux = a;
        a = b;
        b = aux;
        trocou = 1;
    }

    // Agora, fica testando o final. Se a ficar menor que b, sabemos que não tem
    // como b ser segmento de a.
    while (a > b)
    {
        if (verificaFinal (a,b))
        {
            // Achou! Retorna ASEGB ou BSEGA, dependendo se trocamos ou não.
            if (trocou) // b era maior que a -> a era segmento de b!
                return (ASEGB);
            else // a era maior que b -> b era segmento de a!
                return (BSEGA);
        }

        // Não achou, descarta o último dígito de a.
        a /= 10;
    }

    return (0);
}
```