Exercício 1

Valor de a: 10	Valor de b: 20	Valor de c: 30

JUSTIFICATIVA:

Variável A:
A variável A é inicializada juntamente com um ponteiro (*PA). Nesse ponteiro é atribuído o endereço de memória de A. O valor 10 é atribuído à A, que posteriormente é exibido no console.

Variável B:
A variável é inicializada juntamente com os ponteiros *PA e *PB. Nesses ponteiros são atribuídos os endereços de memória de A e B, respectivamente. Logo, vem o motivo de B ser 20, pois B recebe o conteúdo do ponteiro *PA (aponta para o endereço de memória de A, portanto seu conteúdo será o que está contido em A, que é 10) vezes 2. Ou seja, b = 10 * 2 que é 20.

Variável C:
A variável é inicializada juntamente com os ponteiros *PA e *PC. Nesses ponteiros são atribuídos os endereços de memória de A e C, respectivamente. Desta forma, C é igual a 30, pois o ponteiro *PC recebe *PA(10) + b(20) = 30. Mesmo parecendo que C não foi manipulado em nenhum momento, ele foi sim. No momento em que o ponteiro, que aponta para o endereço de memória de C, altera o conteúdo, C também é alterado, já que está armazenado neste mesmo endereço.

Exercício 2

Valor de a: 2	Valor de b: 6	Valor de c: 10

JUSTIFICATIVA:

Variável A:
Nesse cenário, a variável "a" recebe 2 e em uma função é realizada a passagem de parâmetro por valor. Isso quer dizer que "a" irá ser armazenado em um local de memória diferente do parâmetro da função. Desta forma, temos:
p1 = p1 * 2
p1 = a * 2
p1 = 2 * 2
p1 = 4
Vemos que p1 é igual a 4, porém "a" seguirá com seu valor inicial (2), pois como já dito o "p1" armazenou seu conteúdo em um endereço de memória diferente de "a", desta forma não há ligação entre eles.

Variável B:
Nesse cenário, a variável "b" recebe 3 e em uma função é realizada a passagem de parâmetro por referência. Portanto, no momento da passagem de parâmetro, o

"p2" irá receber o endereço de memória relativo à variável "b". Já no momento do cálculo, "p1" modifica o conteúdo daquele endereço. Desta forma, temos:
p2 = p2 * 2;
p2 = 3 * 2
p2 = 6
Vemos que o conteúdo de "p2" é 6. Ao alterar seu valor, será refletido diretamente em "b", pois como já dito ambas estão apontando para o mesmo endereço de memória, que faz com que a alteração seja surtida também em "b". Por isso b também terá o valor 6.

Variável C:
Nesse cenário, a variável "c" recebe 0 e em uma função é realizada a passagem de parâmetro por ponteiro. Por ser uma passagem de parâmetro por ponteiro, há a necessidade de enviar um endereço. Então é enviado o endereço de memória de "c". Na função temos:
*p3 = p1 + p2
*p3 = 4 + 6
*p3 = 10
O conteúdo do endereço que o ponteiro (*p3) aponta é alterado para a soma de "p1" e "p2", resultando em 10. Desta forma, "c" também será 10, pois assim como na passagem por referência, na passagem por ponteiros o conteúdo do endereço de memória da variável "c" será alterado.

Exercício 3

Valor de a: 2	Valor de b: 6	Valor de c: 5

JUSTIFICATIVA:

Variável A:
Nesse cenário, a variável "a" recebe 2 e em uma função é realizada a passagem de parâmetro por valor. Isso quer dizer que "a" irá ser armazenado em um local de memória diferente do parâmetro da função. Desta forma, temos:
p1 = p1 * 2
p1 = a * 2
p1 = 2 * 2
p1 = 4
Vemos que "p1" é igual a 4, porém "a" seguirá com seu valor inicial (2), pois como já dito o "p1" armazenou seu conteúdo em um endereço de memória diferente de "a", desta forma não há ligação entre eles.

Variável B:
Nesse cenário, a variável "b" recebe 3 e em uma função é realizada a passagem de parâmetro por ponteiro. Por ser uma passagem de parâmetro por ponteiro, há a

necessidade de enviar um endereço. Então é enviado o endereço de memória de "b". Na função temos:
*p2 = *p2 * 2
*p2 = 3 * 2
*p2 = 6
O conteúdo do endereço de memória que o ponteiro (*p2) aponta é alterado para o produto dele mesmo vezes 2, resultando em 6. Desta forma, "b" também será 6, pois assim como na passagem por referência, na passagem por ponteiros o conteúdo do endereço de memória da variável "b" será alterado.

Variável C:
Nesse cenário, a variável "c" recebe 3 e em uma função é realizada a passagem de parâmetro por referência. Portanto, no momento da passagem de parâmetro, o "p3" irá receber o endereço de memória relativo à variável "c". Já no momento do cálculo, "p3" modifica o conteúdo daquele endereço. Desta forma, temos:
p3 = p1 + *p2;
p3 = 2 + 3
p3 = 5
Vemos que o conteúdo de "p3" é 5. Ao alterar seu valor, será refletido diretamente em "c", pois como já dito ambas estão apontando para o mesmo endereço de memória, que faz com que a alteração seja surtida também em "b". Por isso b também terá o valor 5.

Exercício 4

Valor de a: 2	Valor de b: 6	Valor de c: 5

JUSTIFICATIVA:

Neste cenário, é instanciado um ponteiro de vetor com 5 inteiros. Assim, será alocado memória para esses 5 inteiros (4*5=20 bits).
Ao passar o vetor como parâmetro na função "enche", será enviado o endereço da primeira posição do vetor. Já no laço de repetição for, o conteúdo do ponteiro será incluído, sendo que a cada iteração, o endereço onde será armazenado irá ser incrementado, ou seja, passará para a próxima casa de memória (de 4 em 4 bits). Permitindo assim, a inserção de novos dados nos endereços posteriores. Exemplo:

Conteúdo	10	20	30	40	50
Endereço memória	c32f48 →	c32f52 →	c32f56 →	c32f60 →	c32f64

Isso acontecerá até o vetor se preencher por completo. Sendo assim, temos:
*v = [10, 20, 30, 40, 50]

Após a execução da função "enche", será utilizado outro laço for, para incrementar no endereço onde está armazenado o vetor (que por padrão é o primeiro). Ele irá fazer com que se pule duas casas de memória e seja equivalente a segunda posição de vetor.

Saída 1:

Ao se deparar com "*v", seria retornado, por padrão, o conteúdo da primeira posição do vetor. Porém, devido ao for anterior ter modificado o endereço de memória para duas casas seguintes, o valor que será exibido será 30.

Conteúdo	10	20	30	40	50
Endereço memória	c32f48 →	c32f52 →	c32f56 →	c32f60 →	c32f64

Saída 2:

Já na saída 2, será exibido o conteúdo do vetor na posição 2, porém ela começa a contar a partir da posição de memória atual.

			0	1	2
Conteúdo	10	20	30	40	50
Endereço memória	c32f48 →	c32f52 →	c32f56 →	c32f60 →	c32f64