Lista de exercícios
– Alocação Dinâmica de Memória –

1. Encontre e corrija o(s) erro(s) do programa a seguir:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int i, p[5];
6      p = calloc(5, sizeof(char));
7
8      for(i = 0; i < 5; i++) {
9          (p + i) = (*p) + i + 1;
10     }
11 }
```

2. Explique, com suas palavras, por que no exercício anterior foi utilizado `calloc` ao invés de `malloc` para alocação de memória.

3. O que deve ser inserido nas lacunas das linhas 8, 17 e 18, respectivamente, para que o programa faça uso das boas práticas de alocação de memória?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int *p, i;
6      p = malloc(sizeof(int) * 9);
7
8      if(_______________) {
9          printf("Out of Memory!");
10         exit(0);
11     }
12
13     for(i = 0; i < 9; i++) {
14         *(p + i) = (i * i) + 24;
15     }
16
17     _______;
18     _______;
19 }
```

   (a) `p = NULL; free(p); p = NULL`

   (b) `p == NULL; free(p); p = NULL`

   (c) `p == NULL; p = NULL; free(p)`

   (d) `p = NULL; p = NULL; free(p)`

4. No exercício a seguir, preencha as lacunas a fim de fazer com que o programa armazene em um vetor de 15 posições o valor de cada posição elevado à terceira potência. E.g.: Se `i = 2`, então `v[i] = 8`. Use alocação dinâmica de memória, manipulação de ponteiros e boas práticas.

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

int main() {
    int *p, i;
    p = ---------------;

    for(i = 0; i < 15; i++) {
        ---------- = i*i*i;
    }

    -------;
    -------;
}
```

5. Adapte o programa do exercício 4 para que, ao final, imprima o valor da posição 11 do vetor.

6. Preencha as lacunas no exercício a seguir a fim de fazer com que o vetor cresça apenas o necessário. Use alocação de memória, manipulação de ponteiros e boas práticas.

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

int main() {
    int i = 0, j = 1, *ptr;
    ptr = ---------(sizeof(int) * (j + 1));

    printf("Use 0 para sair do programa.\n");

    do
    {
        printf("Digite um numero: ");
        scanf("%d", &i);

        if(i != 0)
        {
            ptr = ---------(ptr, sizeof(int) * (j + 1));
            *(ptr + (j - 1)) = i;
            j++;
        }
    }
    while(i != 0);

    for(i = 0; i < (j - 1); i++)
        printf("Valor %d: %d\n", i, ---------);

    ---------;
    ---------;
}
```

Respostas:
Exercício 1:
Linha 5: `p` deve ser um ponteiro.
Correto: `int i, *p;`
Linha 6: O tipo de dado deve ser `int`.
Correto: `p = calloc(5, sizeof(int));`
Linha 9: `(*p) + i + 1` deve ser atribuído ao conteúdo do endereço de memória *p+i*.
Correto: `*(p + i) = (*p) + i + 1;`

Exercício 2:
A função `malloc` faz a alocação da memória sem definir um valor inicial para os endereços alocados, enquanto a função `calloc` inicializa-os todos com o valor nulo (ou zero). Utilizar `malloc` poderia fazer com que um valor aleatório fosse usado ao operar na linha 9, uma vez que o valor do primeiro elemento (*p) não foi definido.

Exercício 3:
Alternativa B

Exercício 4:
Linha 6: `malloc(sizeof(int) * 15)`
Linha 9: `*(p + i)`
Linha 12: `free(p)`
Linha 13: `p = NULL`

Exercício 5:
A linha `printf("Valor Pos. 11: %d", *(p + 11));` deve ser inserida entre as linhas 10 e 12.

Exercício 6:
Linha 6: `malloc`
Linha 17: `realloc`
Linha 25: `*(ptr + i)`
Linha 27: `free(ptr)`
Linha 28: `ptr = NULL`