01. O que imprime o programa a seguir? Tente entendê-lo e responder. A seguir, execute-o e comprove o resultado.

```
int main()
{
  int t, i, M[3][4];
  for (t=0; t<3; ++t)
    for (i=0; i<4; ++i)
      M[t][i] = (t*4)+i+1;
    for (t=0; t<3; ++t)
    {
      for (i=0; i<4; ++i)
        printf ("%3d ", M[t][i]);
      printf ("\n");
    }
  return(0);
}
```

02. Responda as perguntas a seguir. Se achar necessário, faça testes no programa em C.

a) Explique a diferença entre

`p++;` `(*p)++;` `*(p++);`

b) O que quer dizer `*(p+10);`?

03. Qual o valor de `y` no final do programa? Tente primeiro descobrir e depois verifique no computador o resultado. A seguir, escreva um `/* comentário */` em cada comando de atribuição explicando o que ele faz e o valor da variável à esquerda do '=' após sua execução.

```
int main()
{
  int y, *p, x;
  y = 0;
  p = &y;
  x = *p;
  x = 4;
  (*p)++;
  (*p) += x;
  printf ("y = %d\n", y);
  return(0);
}
```

04. Reescreva o programa abaixo usando ponteiros.

```
int main ()
{
  float matrx [50][50];
  int i,j;
  for (i=0;i<50;i++)
    for (j=0;j<50;j++)
      matrx[i][j]=0.0;
  return(0);
}
```

05. Diga quais expressões abaixo são válidas ou não. Considere as declarações

```
int vetor[10];
int *ponteiro;
```

a) `vetor = vetor + 2;`

b) `vetor++;`

c) `vetor = ponteiro;`

d) `ponteiro = vetor;`

e) `ponteiro = vetor + 2;`

06. Explique o que faz o programa abaixo

```
main(){
  float vet[5] = {1.1,2.2,3.3,4.4,5.5};
  float *f;
  int i;
  f = vet;
  printf("contador/valor/valor/endereco/endereco");
  for(i = 0 ; i <= 4 ; i++){
    printf("\ni = %d",i);
    printf("   vet[%d] = %.1f",i, vet[i]);
    printf("   *(f + %d) = %.1f",i, *(f+i));
    printf("   &vet[%d] = %x",i, (int)&vet[i]);
    printf("   (f + %d) = %x",i, (int)(f+i));
  }
}
```

07. Assumindo que `M1[]` é um vetor do tipo `int`, quais das seguintes expressões referenciam o valor do terceiro elemento de `M1`?

a) `*(M1 + 2)`

b) `*(M1 + 4)`

c) `M1 + 4`

d) `M1 + 2`

08. Considere a declaração:

```
int mat[4], *p, x;
```

Quais expressões são válidas? Justifique:

a) `p = mat + 1;`

b) `p = mat++;`

c) `p = ++mat;`

d) `x = (*mat)++;`

09. Faça um programa usando ponteiros, para ordenar 5 números e mostrá-los ordenados na tela.

10. Explique o que o programa abaixo faz:

```
main() {
  int vet[] = {4,9,12};
  int i,*ptr;
  ptr = vet;
  for(i = 0 ; i < 3 ; i++)
    printf("%d ",*ptr++);
}
```

11. Seja `vet` um vetor de 4 elementos: `TIPO vet[4]`. Supor que depois da declaração, `vet` seja armazenado no endereço de memória 4092 (ou seja, o endereço de `vet[0]`). Supor também que na máquina usada uma variável do tipo `char` ocupa 1 byte, do tipo `int` ocupa 2 bytes, do tipo `float` ocupa 4 bytes e do tipo `double` ocupa 8 bytes.

Qual o valor de `vet+1`, `vet+2` e `vet+3` se:

a) `vet` for declarado como `char`?

b) `vet` for declarado como `int`?

c) `vet` for declarado como `float`?

d) `vet` for declarado como `double`?

12. Escreva um programa em C que declare um vetor de 100 elementos inteiros. Você deve inicializar o vetor com zeros usando ponteiros para endereçar seus elementos. Preencha depois o vetor com os números de 1 a 100, também usando ponteiros.

Observação: é expressamente proibido a utilização de qualquer função da biblioteca `string.h` ou de qualquer outra biblioteca em C que faça a manipulação de *strings*, nos exercícios 13 a 22.

13. Implemente em C a função `int equalsMy(char *str1, char *str2)` que retornará `1` caso as duas *strings* sejam idênticas ou `0` caso contrário. Apresente também um programa de testes para validar a função desenvolvida.

14. Implemente em C a função `int equalsIgnoreCaseMy(char *str1, char *str2)` que retornará `1` caso as duas *strings* sejam idênticas, independentemente de estarem em minúsculas ou maiúsculas, ou `0` caso contrário. Apresente também um programa de testes para validar a função desenvolvida.

15. Implemente em C a função `int endsWithMy(char *str, char *suf)` que retornará `1` caso a *string* `str` termine com o sufixo `suf`, ou `0` caso contrário. Apresente também um programa de testes para validar a função desenvolvida.

16. Implemente em C a função `int startsWithMy(char *str, char *pre)` que retornará `1` caso a *string* `str` comece com o prefixo `pre`, ou `0` caso contrário. Apresente também um programa de testes para validar a função desenvolvida.

17. Implemente em C a função `int substringMy(char *str, char *sub)` que retornará `1` caso a *string* `str` contenha a *substring* `sub`, ou `0` caso contrário. Apresente também um programa de testes para validar a função desenvolvida.

18. Implemente em C a função `char *reverseMy(char *str)` que retornará um ponteiro para a *string* `str` invertida. Apresente também um programa de testes para validar a função desenvolvida.

19. Implemente em C a função `char *replaceMy(char *str, char *sub, char *pattern)` que retornará um ponteiro para a *string* `str`, que terá a primeira ocorrência da *substring* `sub` substituída pela *string* `pattern`. Observação: considere que `sub` e `pattern` tenham o mesmo número de caracteres. Apresente também um programa de testes para validar a função desenvolvida.

20. Implemente em C a função `char *replaceAllMy(char *str, char *sub, char *pattern)` que retornará um ponteiro para a *string* `str`, que terá todas as ocorrências da *substring* `sub` substituídas pela *string* `pattern`. Observação: considere que `sub` e `pattern` tenham o mesmo número de caracteres. Apresente também um programa de testes para validar a função desenvolvida.

21. Implemente em C a função `char *indexOfMy(char *str, char ch)` que retornará um ponteiro para a primeira ocorrência do caractere `ch` na *string* `str`. Apresente também um programa de testes para validar a função desenvolvida.

22. Implemente em C a função `char *lastIndexOfMy(char *str, char ch)` que retornará um ponteiro para a última ocorrência do caractere `ch` na *string* `str`. Apresente também um programa de testes para validar a função desenvolvida.

23. Desenvolva um programa em C que faça o controle do estoque de um supermercado, armazenando os produtos e as quantidades que se encontram no estoque no momento. O programa deverá fornecer as seguintes funcionalidades:

a) incluir um novo produto ao estoque, verificando se o mesmo já não se encontra cadastrado;
b) retirar um produto do estoque, verificando se a quantidade do referido produto é zero.
c) adicionar uma quantidade ao produto em estoque, considerando que o produto já se encontra cadastrado;
d) retirar uma quantidade do produto do estoque, verificando se a quantidade solicitada para a retirada seja maior ou igual a quantidade em estoque;
e) imprimir a lista dos produtos e as suas respectivas quantidades em estoque.

Sugestões de implementação:
a) utilize dois vetores para armazenar os valores, um vetor de string para os produtos e um vetor de inteiros para as quantidades, utilizando o mesmo índice nos dois para obter o produto e a sua quantidade;
b) fixe um limite máximo de produtos a serem cadastrados, como por exemplo, 100;
c) converta o nome dos produtos para um padrão definido, como por exemplo, para maiúsculo, para que produtos escritos de forma diferente, sejam referenciados como o mesmo produto. Exemplo: bolacha = Bolacha = BOLACHA