1 Indique verdadeiro ou falso

1.1. O operador & permite-nos obter o endereço de uma variável. Permite também obter o endereço de um ponteiro.

R: Verdadeiro

1.2. O endereço de uma variável que ocupa mais do que um byte de memória é o seu menor endereço.

R: Verdadeiro

1.3. Se x é um inteiro e ptr um ponteiro para inteiros e ambos contêm no seu interior o número 100, então x+1 e ptr+1 apresentarão o número 101.

R: Verdadeiro;

```
#include <stdio.h>
int main(){
        int x, *ptr;
        x = *ptr = 100;
        printf("x %d |ptr  %d ", x + 1, 1 + *ptr);
        return 0;}
```

1.4. o operador asterisco (apontador) permite saber qual o valor de um ponteiro.

R: Verdadeiro

2.

R:

	Usando S	Usando ptr	Valor
	s[2]	*(ptr+2)	**'r'**
	&(s[0])	&(*(ptr+0))	**1000**
	&(s[1])	&(*(ptr+1))	**1001**
	&(s[7])	&(*(ptr+7))	**1007**

3. Qual operador nos permite obter o endereço de uma variável?
R: &

4. Qual caractere que se coloca na declaração duma variável para indicar que ela é um ponteiro?
R: *

5.Onde se coloca este caractere?
R: ? é usado numa condição ou se estiver se referindo a pergunta 4.: é colocado num variável para declarar um ponteiro ou para colocar o valor da variável que está no ponteiro como num print, ex *ponteiro

6.O que contém uma variável do tipo ponteiro?

R: há um endereço

7.Qual o operador que permite saber o que se encontra na posição de memória armazenada num ponteiro?

R: *

8.Se ptr for um ponteiro, qual o valor de *(&ptr)?

R: “será uma variável”, determinada seu valor ou não

9.Como declaramos uma variável que tenha capacidade de conter o endereço de um ponteiro para float?

R: float *ponteiro;

10.faça um programa que crie um vetor de 10 inteiros, coloque valores nele e depois imprima todos os seus conteúdos. Impressão dos conteúdos deverá ser feita num ponteiro?

R:

```c
#include <stdio.h>
int main()
{
  int n[10];
  int *ponteiro;
  n[0] = 0;
  n[1] = 1;
  n[2] = 4;
  n[3] = 9;
  n[4] = 16;
  n[5] = 25;
  n[6] = 36;
  n[7] = 49;
  n[8] = 64;
  n[9] = 81;
  ponteiro = &n;
  for(int i = 0 ; i < 10; i++)
  {
    printf("\n%d;", *(ponteiro + i));
  }
  return 0;
}
```

feito no repl com padrão atual em c