INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA – CAMPUS CAMPINA GRANDE			
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PARAÍBA Campus Campina Grande	CURSO:	CURSO ENGENHARIA DA COMPUTAÇÃO	
	PERÍODO:		TURMA:
	DISCIPLINA:	PROGRAMAÇÃO E ESTRUTURAS DE DADOS	
	PROFESSOR:	CÉSAR ROCHA VASCONCELOS	SEMESTRE LETIVO

Lista 01 - Apontadores

1) Considere o código abaixo escrito em C e responda as questões abaixo: (procure não utilizar o compilador)

```
int a=5, b=6, c=7;
int v[10] = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 };
int * pt1, * pt2, * pt3;

pt1= &a;
pt2= &b;
pt3= &c;

pt1= pt3;
pt2= pt3;
*pt1 = *pt2 + *pt3;

pt1= &v[3];
for (int k=0; k<3; k++) {
  *pt1 = *pt1 + 1000;
  pt1 = pt1 + 2;
}
```

a) Verifique se há erro(s) no código. Caso afirmativo, corrija-os.
b) Qual o conteúdo final das variáveis a, b e c?
c) Qual o conteúdo final armazenado no vetor v?

2) Explique, com suas palavras, o que significa cada uma das instruções em C abaixo:
a) p = nomeVetor e p = &nomeVetor[3]
b) p++ e *p++
c) (*p)++ e *(p++)
d) *(p + 8) e (*p) + 8
e) *(p + 2) e p[2]
f) if (ptl == (pt2 + 6)) ...

3) Considere o código abaixo escrito em C e responda as questões seguintes: (procure não utilizar o compilador)

```
int a=5, b=6, c=7;
int v[10] = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 };
int * pt1, * pt2, * pt3;

pt1= &a;
pt2= &b;
pt3= &c;

pt2= pt1;
*pt3 = *pt2 + 2000;

pt1= &v[8];
for (int k=0; k<2; k++) {
  *pt1 = *pt1 - 5;
  pt1 = pt1 - 3;
}
```

a) Verifique se há erro(s) no código. Caso afirmativo, corrija-os.
b) Qual o conteúdo final das variáveis a, b e c?
c) Qual o conteúdo final armazenado no vetor v?

4) Verifique se há erros nos dois trechos de código em C abaixo (um em cada coluna). Depois, indique quais as saídas dos mesmos (procure não utilizar o compilador).

```
int main() {
  int y, *p, x;
  y = 0;
  p = &y;
  x = *p;
  x = 4;
  (*p)++;
  x--;
  (*p) += x;
  y = (*p)+ 10;
  printf ("y = %d\n", y);
  printf ("x = %d\n", x);
  printf ("(*p) = %d\n", *p);
  return(0);
}
```

```
#include<stdio.h>
#include<conio.h>
void main(void) {
   int x[3]= {10,20,30};
   int *pt;

   clrscr();
   pt = x;
   pt++;
   printf("\n%d ", *(--pt));
   printf("\n%d ", *pt);
   pt = &x[0];
   printf("\n%d ", *(pt++));
   printf("\n%d ", *pt);
}
```

5) Considere o trecho de código abaixo escrito em C e responda as questões seguintes: (procure não utilizar o compilador)

```
int a=5, b=6, c=7;
int v[10] = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 };
int * pt1, * pt2, * pt3;

pt3= &c;

pt1= pt2;
pt3= pt2;
*pt2 = *pt3 + 1000;

pt1= v;
pt1 = pt1 + 1;
for (int k=0; k<2; k++) {
  *pt1 = *pt1 / 2;
  pt1 = pt1 + 4;
}
```

a) Verifique se há erro(s) no código. Caso afirmativo, corrija-os.
b) Qual o conteúdo final das variáveis a, b e c?
c) Qual o conteúdo final armazenado no vetor v?

6) Escreva uma função `char *imprimeEstadoCivil(int codigo)` em C que, com base num código fornecido como parâmetro, possa devolver um ponteiro para a cadeia de caracteres correspondente: Solteiro(a), Casado(a), Divorciado(a) e Viúvo(a). Utilize o conceito de variáveis locais estáticas para que as cadeias não sejam destruídas no término da função. Ao final, crie um programa (com um *main()* simples) para validar/testar sua função.

7) Escreva uma função `void imprimeVetor(char *umVetor)` em C que possa receber um vetor e imprimí-lo usando apontadores (em vez de indexação simples). Ao final, crie um programa (com um *main()* simples) para validar/testar sua função.

8) Escreva uma função `void toUpperCase (char *umVetor)` em C que possa receber um vetor de caracteres e alterar todos os seus caracteres para maiúsculos. Dica: utilize a função `toupper` na biblioteca `ctype.h`. Ao final, crie um programa (com um *main()* simples) para validar/testar sua função.

9) Escreva uma função `char * inverteVetor(char *umVetor)` em C que receba um vetor de caracteres e possa inverter seu próprio conteúdo. Ex: vl = {'s', 'o', 'l' ,'\0'} retornaria vl = {'l', 'o', 's' ,'\0'}. Ao final, crie um programa (com um *main()* simples) para validar/testar sua função.

10) Escreva uma função `int strEnd(char *strOrigem, char *strFim)` em C que possa verificar se a *String* `strFim` se encontra no final da cadeia `strOrigem`. A função deve retornar `1` caso verdadeiro e `0` caso contrário. Ao final, crie um programa (com um *main()* simples) para validar/testar sua função.