Universidade Federal do ABC
Bacharelado em Ciência da Computação - 27/10/2017
Prova I de Programação Estruturada - Turma A1

NOME/RA : .....

Instruções

1. Escreva com caneta o seu nome completo e o RA;
2. As respostas devem ser transcritas com caneta esferográfica;
3. Cada questão vale 03 (três) pontos;
4. 01 ponto será dado caso siga os padrões descritos em sala de aula;
5. As questões serão corrigidas considerando corretude, padronização e estruturação do código.

Questões

Questão 1. Estruture o código abaixo de acordo com os conceitos aprendidos na disciplina [2,0 pts] e otimize-o [1,0 pt]:

```c
#include <stdio.h>

/* Persistencia aditiva de um numero eh quantas vezes
 * podemos somar os digitos de um numero ate que ele
 * tenha apenas um digito.
 */
int  persistencia_aditiva(int n)
{
  unsigned long long  novo_n, quantas = 0;

  while (n>=10) {
    novo_n = 0;
    while (n != 0) {
      novo_n += n%10;
      n = n/10;
    }
    quantas = quantas + 1;
    n = novo_n;
  }
  return quantas;
}
int main()
{
  printf("%d\n", persistencia_aditiva(199));
  return 0;
}
```

```c
#include <stdio.h>

int soma_digitos (int x)
{
  int soma = 0;

  /* se eu colocar x != 0, farei uma opera  o de / e % a toa */
  while (x >= 10) {
    soma += x%10;
    x = x/10;
  }

  return soma + x; /* como agora eh x >= 10, devo somar x ao final */
}
```

```c
/* Persistencia aditiva de um numero eh quantas vezes
 * podemos somar os digitos de um numero ate que ele
 * tenha apenas um digito.
 */
int  persistencia_aditiva(int n)
{
  int quantas = 0;

  while (n>=10) {
    n = soma_digitos(n);
    quantas++;
  }

  return quantas;
}

int main()
{
  printf("%d\n", persistencia_aditiva(199));
  return 0;
}
```

Questão 2. O valor de x^n pode ser calculado recursivamente como:

$$x^n = \begin{cases} 1, & \text{para n==0} \\ x, & \text{para n==1} \\ (x^{n/2})^2, & \text{para n par} \\ x \cdot (x^{n/2})^2, & \text{para n impar} \end{cases} \quad (1)$$

Implemente o algoritmo em sua versão de recursão caudal [3,0 pts].

```c
#include <stdio.h>

int pow_n (int x, int n)
{
  int xn2;

  if (n==0) return 1;
  if (n==1) return x;

  xn2 = pow_n(x, n/2);

  if (n%2) return xn2*xn2*x;
  return xn2*xn2;

}

/* leve em conta que (x^(n/2))^2 == (x^2)^(n/2) */
int pow_nTR (int x, int n, int pot)
{

  if (n==0) return pot;
  if (n==1) return x*pot;
  if (n%2) return pow_nTR(x*x, n/2, x*pot);
  return pow_nTR(x*x, n/2, pot);

}

int main ()
{
  printf("%d\n", pow_n(3,6));
  printf("%d\n", pow_nTR(3,6,1));
  return 0;
}
```

Questão 3. Corrija o código abaixo para que retorne o que é esperado [3,0 pts]:

```c
#include <stdio.h>

/* compara duas strings, retorna 0 se forem iguais
  * -1 se s1 for antes de s2, e +1 se s1 for depois de s2
  */
 int strcmp (char *s1, char *s2)
 {
     /* enquanto s1 nao chegou ao fim e
      * o caracter atual de s1 for igual a s2
      */
     while (*s1 != '\0' && s1 == s2) {  /* correcao: *s1 == *s2 */
         s1++;
         s2++;
     }

     if (*s1 < *s2) return -1;
     if (*s1 > *s2) return 1;
     return 0;

 }

int main(void)
{
  char s1[100] = "Ola\0jasdakjshdkjahsd";
  char s2[100] = "Ola Mundo\0asdafasdas";

  printf("s1 = %s\ns2 = %s\n", s1, s2);

  printf("s1 == s2? %d\n", strcmp(s1,s2));

  return 0;
}
```