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>

/* Encontre o menor nu  mero natural divisivel pelos numeros de 1 a 20 */
int menor_divisivel_20()
{
  int n, j, ehDivisivel;

  n = 20;

  while(1) {
    ehDivisivel = 1;
    for (j=1; j<=20; j++) {
      if ((n%j)!=0) {
        ehDivisivel = 0;
      }
    }
    if (ehDivisivel) {
      return n;
    } else {
      n = n + 20;
    }

  }
}

int main()
{
  printf("%d\n", menor_divisivel_20());
  return 0;
}
```

```c
#include <stdio.h>

/* Encontre o menor nu  mero natural divisivel pelos numeros de 1 a 20 */

char eh_divisivel(int x)
{
  int i;

  /* ele sempre sera divisivel por 1, e se for por 20 sera por 2 tb */
  /* para otimizar ainda mais, poderiamos criar uma lista dos testes a serem feitos:
```

```
   * [20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 9, 8, 7, 6, 3]
   * N o testamos o 20 pois iremos passar apenas multiplos de 20,
   * comecamos pelo 19 pois    o mais provavel de falhar.
  */
  for (i=19; i>=3; i--) {
    if ((x%i) != 0) return 0;
  }
  return 1;
}

int menor_divisivel_20()
{
  int n = 20;

  while(!eh_divisivel(n)) {
    n += 20;
  }

  return n;
}

int main()
{
  printf("%d\n", menor_divisivel_20());
  return 0;
}
```

Questão 2. O comprimento da sequência de Collatz de um número n pode ser calculada pela relação recursiva:

$$C(n) = \begin{cases} 1, & \text{para n==1} \\ 1 + C(n/2), & \text{para n par} \\ 1 + C(3n+1), & \text{para n impar} \end{cases} \quad (1)$$

```
#include <stdio.h>

int collatz(int n)
{
  if (n==1) return 1;
  if (n%2)  return 1 + collatz(3*n + 1);
  return 1 + collatz(n/2);
}

int collatzTR(int n, int length)
{
  if (n==1) return 1 + length;
  if (n%2)  return collatzTR(3*n + 1, length + 1);
  return collatzTR(n/2, length + 1);
}

int main ()
{
  printf("%d\n", collatz(10));
  printf("%d\n", collatzTR(10, 0));
  return 0;
}
```

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

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

```
#include <stdio.h>

/* concatena duas strings */
 char * strcat (char * s1, char * s2)
 {
     /* nao vamos perder o ponteiro original */
     char *s = s1;

```

```
    /* chega ao final do ponteiro */
    while (*s != '\0')
        s++;

    /* passa por cima do \0 de s1 e copia s2 */
    while (*s2 != '\0') {
        *s = *s2;
        s2++;
        s++;
    }

    *s2 = '\0';  /* incluir essa linha para correcao */

    return s1;
}

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

  return 0;
}
```