Prova 1

Algoritmos e Estruturas de Dados I

Professores: Ítalo S. Cunha e Pedro O.S. Vaz de Melo

8 de outubro de 2015

Nome: _____

escrevendo o meu nome eu juro que seguirei o código de honra

Código de Honra para este exame:

- Não darei ajuda a outros colegas durante os exames, nem lhes pedirei ajuda;
- não copiarei nem deixarei que um colega copie de mim;
- não usarei no exame elementos de consulta não autorizados.

Informações importantes:

- Em questões que pede um **programa**, este deve ser completo, com bibliotecas (incluindo, quando necessário, a biblioteca `prova1.h`), função main, etc. Se deve ser feita uma **função**, somente a função é suficiente. Se deve ser feito um **procedimento**, somente o procedimento é suficiente.
- A interpretação das questões da prova faz parte do critério de avaliação. Caso tenha dúvida sobre a sua interpretação de uma determinada questão, escreva as suas suposições na resolução da mesma.
- As funções implementadas no módulo `prova1.h` podem ser usadas em **qualquer** exercício da prova. Além disso, se você usar uma função do módulo `prova1.h`, considere que ela está implementada de forma correta.

Referências:

Função/Operador	Descrição	Biblioteca	Exemplo
`%`	retorna o resto da divisão	-	`20 % 3` retorna 2

1. (*6 points*) Implemente funções para realizar as operações abaixo sobre parâmetros recebidos como números inteiros sem sinal (`unsigned int`). Suas funções **não devem** usar condicionais (`if`). Dica: algumas delas podem requerer operações bit-a-bit. Abaixo um exemplo de uma função que retorna o negativo do parâmetro:

```
int neg(unsigned int number) {
   return -number;
}
```

a. (*2 pts*) Extrair código de área de números de telefone com 8 dígitos (e.g., para o telefone 3134095858 a sua função deve retorna 31).

```
return number/10000000;
```

b. (*2 pts*) Transformar um número par no próximo número ímpar e manter um número ímpar inalterado (e.g., para o número 4 a sua função deve retornar 5 e para o número 5 a sua função deve retornar 5).

```
return number | 1;
```

c. (*2 pts*) Retornar verdadeiro se o número for par ou falso caso contrário. Dica: lembre dos conceitos de verdadeiro e falso para a linguagem C.

```
return !(number%2);
```

2. (*3 points*) Escreva um **procedimento** de nome `divisao` que recebe como parâmetro dois endereços de memória que armazenam inteiros (ponteiros para inteiros) `end_var1` e `end_var2`. A função deve fazer a divisão do inteiro armazenado em `end_var1` pelo inteiro armazenado em `end_var2`. Depois disso, deve armazenar em `end_var1` o valor da divisão e em `end_var2` o resto da divisão.

```
void divisao(int *endvar1, int *endvar2) {
  int a = *endvar1, b= *endvar2;
  *endvar1 = a/b;
  *endvar2 = a%b;
}
```

3. (*3 points*) Escreva um **programa** para ler 10 pares de números **maiores que zero** (`numerador`,`denominador`) e imprimir o valor e o resto da divisão do `numerador` pelo `denominador` para cada um dos 10 pares. Você **DEVE** fazer uso da função `divisao` do exercício anterior. Você também deve **garantir** que todas as divisões sejam feitas entre números maiores que zero.

```
#include <stdio.h>
#include "prova1.h" //ou <prova1.h>
void main() {
   int i=0, a, b;
   while(i<10) {
      scanf("%d %d", &a, &b);
      if(a>0 && b>0) {
         divisao(&a, &b);
         printf("\n%d %d", a, b);
         i++;
      }
   }
}
```

4. (*5 points*) Escreva uma **programa** para controlar um semáforo de trânsito. Sua função deve utilizar três variáveis inteiras **globais** de nome `verde`, `amarelo` e `vermelho`. Essas variáveis deverão armazenar 1 para indicar se o sinal de uma determinada cor está aceso ou 0 para indicar se está apagado. Suponha também que há uma biblioteca de nome `semaforo.h` que disponibiliza a função `void acende(int segundos)`, que trava a execução do programa por uma quantidade de segundos e, ao mesmo tempo, acende as luzes do semáforo de acordo com as variáveis globais `verde`, `amarelo` e `vermelho`. Exemplo: Se `verde=0`, `amarelo=1` e `vermelho=0`, a execução de `acende(10)` trava o programa por 10 segundos e acende a luz amarela do semáforo e apaga, caso estejam acesas, as luzes verde e vermelha. Assim, o seu programa deve **repetir o seguinte ciclo indefinidamente**: manter o semáforo 60 segundos na luz verde, depois 5 segundos na luz amarela e, depois disso, 45 segundos na luz vermelha.

```
#include <semaforo.h>

void main() {
green = 1;
yellow = 0;
red = 0;
while(1) {
     acende(60);
     green = 0;
     yellow = 1;
     acende(5);
     yellow = 0;
     red = 1;
     acende(45);
     red = 0;
     green = 1;
   }
}
```