Prova 1

Algoritmos e Estruturas de Dados I

Professor: Pedro O.S. Vaz de Melo

28 de abril de 2016

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. (*4 points*)
Escreva uma **função** de nome `primosEntreSi` que verifica se dois números inteiros x e y são primos entre si. Se os números forem primos entre si a função deve retornar 1 e, caso negativo, deve retornar 0. Dois números são primos entre si se o máximo divisor comum entre eles for 1. Assim, a função `primosEntreSi` deve retornar 1 se os parâmetros de entrada forem 27 e 32 e 0 se os parâmetros de entrada forem 26 e 32. Essa função deve ter o seguinte protótipo:

```
int primosEntreSi(unsigned int x, unsigned int y);
```

2. (*5 points*) Escreva um **programa** que lê dois números inteiros do teclado: x_{min} e x_{max} e imprime na tela todos os pares **distintos** de números naturais que são primos entre si e que estão entre x_{min} e x_{max}. x_{min} e x_{max} devem ser maiores que zero e x_{min} deve ser maior que x_{max}. O seu programa deve pedir novos valores de x_{min} e x_{max} enquanto o usuário inserir valores inválidos para eles. Exemplo: se $x_{min} = 3$ e $x_{max} = 7$, o programa deve imprimir
$(3,4), (3,5), (3,7), (4,5), (4,7), (5,6), (5,7), (6,7)$. Note que o par $(4,3)$ não deve ser impresso, pois o par $(3,4)$ já foi impresso. Obs: neste programa você não precisa imprimir qualquer mensagem para o usuário, apenas a resposta final.

3. (*4 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.

4. (*3 points*) Complete o programa abaixo, que deve ler 1000 **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 1000 pares. Complete o programa de forma a garantir que todas as divisões sejam feitas entre números maiores que zero.

```
#include <stdio.h>
#include ______________________

void main() {
  int count=0, num, den;
  while(_______________________________) {
    scanf("%d %d", &num, &den);
    if(__________________________) {
      divisao(_____________________);
      printf("\nvalor: %d, resto: %d", _________________________);
      count++;
    }
  }
}
```