Prova 1

Algoritmos e Estruturas de Dados I - turma W

Professor: Pedro O.S. Vaz de Melo

11 de abril de 2013

Nome: _____

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

Código de Honra para este exame (baseado no *Honor Code* da Universidade de Stanford):

- 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.

Referências:

Função/Operador	Descrição	Biblioteca	Exemplo
`float exp(float x)`	retorna e^x	`math.h`	`exp(1)` retorna 2.71828
`float pow(float b, float e)`	retorna b^e	`math.h`	`pow(2,3` retorna $2^3 = 8$
`%`	retorna o resto da divisão	-	`20 % 3` retorna 2

1. (*5 points*) Para as questões a seguir, considere que as implementações serão feitas no módulo `"prova1.h"`.

a. (*3 pts*) Um estatístico lhe procurou pois precisa de uma implementação em C da função densidade de probabilidade da distribuição exponencial, que é a seguinte:

$$f(x, \lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{se } x >= 0 \\ 0 & \text{se } x < 0 \end{cases} \quad (1)$$

Assim, implemente uma **função** de nome `exppdf` que recebe os parâmetros x e λ e retorna $f(x, \lambda)$, todos pontos flutuantes.

b. (*2 pts*) Implemente uma função que recebe dois inteiros e retorna o menor deles. A função deve ter o seguinte protótipo:

```
int menor(int x, int y);
```

2. (*8 points*) Para as questões a seguir, considere que a implementação da letra **a** será feita no módulo `"prova1.h"`. Para a letra **b**, considere que o módulo `"prova1.h"` tem a função `mdc` da letra **a** implementada corretamente.

a. (*5 pts*) Escreva uma **função** que retorna o máximo divisor comum (MDC) entre dois números inteiros. Essa função deve ter o seguinte protótipo:

```
int mdc(int x, int y);
```

b. (*3 pts*) Escreva um **programa** que lê dois números inteiros do teclado e imprime na tela o máximo divisor comum entre eles. Caso o usuário insira um valor menor ou igual a zero, o programa deve informar isso a ele e pedir um novo número. Esse processo deve se repetir enquanto qualquer um dos números lidos seja menor ou igual a zero.

3. (*7 points*) Para as questões a seguir, considere que a implementação da letra **a** será feita no módulo `"prova1.h"`. Para a letra **b**, considere que o módulo `"prova1.h"` tem a função `igualaAoMenor` da letra **a** implementada corretamente.

a. (*4 pts*) Escreva um **procedimento** de nome `igualaAoMenor` que recebe dois endereços de memória de variáveis inteiras `end_var1` e `end_var2`. A função deve verificar quais desses endereços de memória tem o **menor** inteiro armazenado e deve, em sequida, armazenar esse **menor** valor no endereço de memória que tem o **maior** valor armazenado. Assim, no final da execução, os dois endereços `end_var1` e `end_var2` devem conter o mesmo valor, que é o **menor** inteiro armazenado nesses endereços quando essa função foi chamada.

b. (*3 pts*) Complete o código abaixo:

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

void main(void) {
int x,y;
printf("Digite os valores de x e y\n");
scanf(_____________________________);
igualaAoMenor(___________________);
printf("Os novos valores de x e y sao: x=%d e y=%d\n", x, y);
}
```