Prova 3

Algoritmos e Estruturas de Dados I

Professor: Pedro O.S. Vaz de Melo

23 de junho de 2015 (valor: 30 pontos)

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:

- Considere que todos os procedimentos e funções pedidas nesta prova serão implementados no módulo `prova3.h`.
- Em questões que pede um **programa**, este deve ser completo, com bibliotecas (incluindo o módulo `prova3.h` quando necessário), 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.
- Vocês podem utilizar qualquer função pedida na prova em suas questões. Considere que a implementação da função que você está usando está correta.
- Se você entregar a prova em até 1 hora, você ganha 10% da sua nota + 1 ponto.

Referências:

Função/Operador	Descrição	Exemplo
`FILE* fopen(const char *filename, const char *mode)`	abre o arquivo `filename` no modo `mode`	`FILE *temp = fopen("temp.txt", "w");`
`int fclose ( FILE * arq )`	fecha o arquivo `arq`	`fclose(arq);`
`int feof ( FILE * arq )`	verificar se o arquivo `arq` chegou ao fim	`int fim_arq = feof(arq);`
`int fscanf(FILE *arq, const char *format, endereço das variáveis);`	lê dados númericos do arquivo `arq`	`fscanf(arq, "%f", &nota1);`
`int fprintf(FILE *arq, const char *format, valores/variáveis);`	escreve dados no arquivo `arq`	`fprintf (arq, "valor de aux: %d", aux);`
`void* malloc (size_t size);`	aloca um bloco de memória de tamanho `size`, retornando um ponteiro para o início do bloco.	`int *p1 = (int*)malloc(sizeof(int));`
`void free (void *p);`	Desaloca o bloco de memória apontado por `p`.	`free(p);`
`char *strtok (char *str, const char *delimiters)`	Retorna um campo da *string* `str` separado por um dos caracteres contidos em `delimiters`. Se `str` é `NULL`, busca o campo da string usada na chamada anterior.	`char *nome = strtok(buffer, ",");`

1. **(*8 points*)** A função `hotpo(n)` (*Half Or Triple Plus One*) consiste no seguinte: pegue o número n e o divida por 2 se ele for par, ou multiplique por 3 e some 1 caso n seja ímpar. Repita o processo até que n seja 1. A conjectura de Collatz estipula que esse processo sempre converge para 1. Escreva uma função RECURSIVA que retorna o número de elementos gerados pela função hotpo até que n se torne 1. Imprima cada um dos elementos gerados. Exemplo: hotpo(5) retorna 6 e imprime a sequência 5, 16, 8, 4, 2, 1. A sua função deve ter o seguinte protótipo:

```
int hotpo(int n);
```

2. (*8 points*) Escreva uma função de nome `primeiroNome` que recebe como parâmetro uma *string* `nome_completo` contendo o nome completo de uma pessoa e que retorna uma outra *string* contendo o primeiro nome desta pessoa. Ao invés de modificar a *string* `nome_completo`, você deve criar uma outra *string* na memória para receber o primeiro nome da pessoa e, depois disto feito, retornar o ponteiro (isto é, o endereço de memória) para esta nova área. Considere que o primeiro nome de pessoas que têm nome composto (ex: Ana Paula Silva) é o primeiro nome do nome composto (ex: Ana).

3. (*14 points*) Uma imagem pode ser representada por uma matriz $n \times m$ em que cada célula (i, j) é um inteiro que representa uma cor. Por exemplo, se 0 representa a cor branca e 70 a cor vermelha, a matriz 4×5 abaixo ilustra um quadrado com borda vermelha e interior branco:

70	70	70	70	70
70	0	0	0	70
70	0	0	0	70
70	70	70	70	70

Como células vizinhas costumam conter o mesmo valor, uma maneira de reduzir o tamanho de arquivos de imagens neste formato é, para cada linha da matriz e para cada sequência de números iguais, contar quantos são iguais e trocar essa sequência pelo par $\{k, cor\}$, em que k é o tamanho da sequência e *cor* é a cor desta sequência. Assim, para a matriz acima este método de compactação gera a seguinte saída:

```
5 70
1 70 3 0 1 70
1 70 3 0 1 70
5 70
```

Assim, crie um **programa** para ler uma matriz de uma imagem que está armazenada no arquivo `imagem.img` e, depois disso, criar um arquivo de nome `imagem.cmp` contendo essa imagem no formato compactado. Para facilitar, considere que a primeira linha do arquivo `imagem.img` contém os valores n (número de linhas da imagem) e m (número de colunas da imagem).
Se o arquivo de entrada `imagem.img` conter

```
4 5
70 70 70 70 70
70 0 0 0 70
70 0 0 0 70
70 70 70 70 70
```

o arquivo `imagem.cmp` gerado pelo seu programa deve conter

```
5 70
1 70 3 0 1 70
1 70 3 0 1 70
5 70
```