CCM0128 — Computação II

Curso de Ciências Moleculares — Turma 22 — Primeiro Semestre de 2013

Primeira Prova — 11 de abril de 2013

Nome: _____

Assinatura: _____

Nº USP: _____

Instruções:

1. Preencha o cabeçalho acima.
2. Não destaque as folhas deste caderno.
3. A prova tem quatro questões. Antes de começar a trabalhar, verifique se o seu caderno de questões está completo.
4. A prova pode ser feita a lápis. Cuidado com a legibilidade.
5. A prova deve ser resolvida individualmente. Não é permitida a consulta a livros, apontamentos ou colegas.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Nas questões sobre listas encadeadas, use esta definição de `Celula`:

```
typedef struct cel {
    int conteudo;      /* campo de conteúdo */
    struct cel *prox;  /* ponteiro para a próxima célula */
} Celula;
```

8. Nesta prova você pode usar a função `mallocc` quando quiser. Não é necessário escrever a definição dessa função.
9. A duração da prova é **1 hora e 40 minutos.**

Boa prova!

Questão	Valor	Nota
1	1,5	
2	5,0	
3	2,0	
4	1,5	
Total	10,0	

Questão 1 (1,5 pontos)

Considere esta definição do tipo `Horario`:

```
typedef struct {
    int horas;
    int minutos; /* este campo deve conter um inteiro não negativo menor que 60 */
} Horario;
```

Escreva uma função com o seguinte protótipo:

```
Horario intervalo_de_tempo(Horario inicio, Horario fim);
```

A função `intervalo_de_tempo` devolve uma `struct` com o número de horas e o número de minutos entre os horários `inicio` e `fim`. Você pode supor que o horário `fim` é maior ou igual ao horário `inicio`.

Exemplos: O intervalo de tempo entre 15hs18min e 20hs53min é 5 horas e 35 minutos. O intervalo de tempo entre 10hs45min e 19hs12min é 8 horas e 27 minutos.

Questão 2 (2,5 + 2,5 pontos)

(a) Escreva uma função **recursiva** com o seguinte protótipo:

```
Celula *anterior_maximo(Celula *p);
```

Essa função recebe um ponteiro `p` para a célula cabeça de uma lista encadeada com cabeça, e devolve um ponteiro para a célula anterior a uma célula de conteúdo máximo. Se a lista for vazia, a função devolve `NULL`.

(b) Utilizando, obrigatoriamente, a função `anterior_maximo` do item (a), escreva uma função **iterativa** com o seguinte protótipo:

```
void ordena(Celula *ini);
```

Essa função recebe um ponteiro `ini` para a célula cabeça de uma lista encadeada com cabeça. A função rearranja as células dessa lista de modo que ela fique em ordem crescente (`<=`), usando a idéia do algoritmo de ordenação por seleção que estudamos para vetores. Sua função deve selecionar, a cada passo, o elemento máximo ao invés do mínimo. Não aloque novas células nem altere o campo `conteudo` das células da lista `ini`.

Implemente a seguinte idéia para a função `ordena`: construa uma lista encadeada ordenada sem cabeça, cujo início é dado por uma variável apontadora `ini_ord`, retirando da lista apontada por `ini` a célula de conteúdo máximo, e inserindo essa célula na lista `ini_ord`. No final, inclua na lista `ini_ord` a célula cabeça da lista original (`ini`).

Questão 3 (2,0 pontos)

Considere o arquivo `pilha.h`, cujo conteúdo é o seguinte:

```
void inicia_pilha();    /* inicializa uma pilha */
int  pilha_vazia();     /* verifica se a pilha esta vazia (devolve 0 ou 1) */
void empilha(int item); /* coloca um int na pilha */
int desempilha();       /* retira um int da pilha */
void finaliza_pilha();   /* libera a memória usada pela pilha  */
```

Esse arquivo define a interface de um tipo de dados abstrato para uma pilha de inteiros. Note que o tipo de dados não é de primeira classe: como as funções da interface não recebem um parâmetro especificando a pilha sobre a qual se deseja operar, o pressuposto é que só há uma instância do tipo pilha. As funções da interface manipulam essa pilha.

O objetivo desta questão é implementar esse tipo de dados abstrato usando uma lista encadeada sem cabeça. Em outras palavras, deseja-se criar um arquivo `pilha.c` que implemente a pilha como uma lista encadeada sem cabeça. Para uma lista encadeada funcionar como uma pilha, todas as inserções e remoções de elementos devem ser efetuadas no início da lista.

As linhas iniciais do arquivo `pilha.c` são dadas abaixo. Sua tarefa é completar esse arquivo, escrevendo as funções cujos protótipos estão em `pilha.h`. Se necessário, use o verso desta folha.

```
#include <stdio.h>
#include <stdlib.h>
#include "pilha.h"

typedef struct cel {
    int conteudo;       /* campo de conteúdo */
    struct cel *prox;   /* ponteiro para a próxima célula */
} Celula;

static Celula *topo;    /* ponteiro para a primeira célula da lista */

/* implementações das funções da interface pilha.h */
...
```

Questão 4 (1,5 pontos)

Escreva uma função com o seguinte protótipo:

```
void imprime_pares_com_soma(int soma, int n, int v[]);
```

Essa função recebe um inteiro `soma`, um inteiro não negativo `n` e um vetor estritamente crescente de inteiros `v[0..n-1]`. Ela imprime a lista de todos os pares de elementos distintos do vetor `v` cuja soma é igual ao valor do parâmetro `soma`. Em outras palavras, a função imprime a lista de todos os pares

```
v[i]  v[j]
```

tais que $0 \leq$ `i` $<$ `j` $<$ `n` e `v[i] + v[j] == soma`. Por exemplo: dado o vetor

`a` =

0	1	2	3	4	5	6	7	8	9	10
−5	−3	2	4	5	6	7	9	10	12	15

a chamada `imprime_pares_com_soma(12, 11, a)` imprime a lista de pares

```
-3 15
2 10
5 7
```

Sua função deve consumir tempo $O($`n`$)$. Este requisito é **essencial**.

Rascunho (não destacar esta página do caderno de questões)