Departamento de Informática - PUC-Rio
INF 1007 – Programação 2
P3 – 23/06/2010

Aluno: _____

Matrícula: _____ Turma: _____

Instruções:

1) Escreva seu nome completo, matrícula e turma em todas as folhas desta prova;
2) A prova deve ser completamente resolvida nas folhas que constam deste caderno, utilizando-se frente e/ou verso;
3) As questões podem ser resolvidas em qualquer ordem;
4) As soluções que não forem apresentadas nas páginas a elas destinadas devem ser identificadas com o número da questão a que se referem;
5) A prova pode ser feita utilizando-se lápis ou caneta (azul ou preta);
6) Todos os dispositivos eletrônicos (celulares, i-pods, etc) devem ser desligados.

Pontuação:

Questão	Item	Valor	Nota
1	A	2,0	
	B	2,0	
2	–	3,0	
3	–	3,0	
Total		10,0	

INF 1007 – P3 – 23/06/10	
Nome:	
Matrícula:	Turma:

Questão 1

Os alunos de uma disciplina estão organizados em uma lista encadeada, onde cada elemento da lista é definido pela estrutura a seguir:

```
struct elemento {
   Aluno* info;
   struct elemento* prox;
};
typedef struct elemento Elemento;
```

O conteúdo de cada elemento da lista é um ponteiro para o tipo estruturado *Aluno*, que descreve alunos de um determinado curso, definido a seguir:

```
struct aluno {
   char nome[81];
   char turma;
   float p1, p2, p3;
};
typedef struct aluno Aluno;
```

A) [Valor: 2,0 pontos] Implemente uma função que insere um novo aluno em uma lista, mantendo a lista ordenada alfabeticamente pelo nome dos alunos. A função recebe como parâmetros o ponteiro *lst*, para o início da lista, e o ponteiro *novo*, para o tipo estruturado *Aluno,* que representa o novo aluno, e deve retornar um novo ponteiro para o início da lista, obedecendo ao seguinte protótipo:

```
Elemento* lst_insere_aluno(Elemento* lst, Aluno* novo);
```

B) [Valor: 2,0 pontos] Implemente uma função que retorna uma nova lista contendo somente os alunos aprovados que pertencem a uma determinada turma. Para ser aprovado, o aluno deve ter a média aritmética das notas da P1, P2 e P3 maior ou igual a 5. Na nova lista retornada, os alunos devem aparecer em ordem alfabética de nome. A lista original não pode ser alterada. A função recebe como parâmetros o ponteiro *lst*, para o início da lista, e o caractere *turma*, que indica a turma desejada, e deve retornar o ponteiro para o início da nova lista, obedecendo ao seguinte protótipo:

```
Elemento* lst_aprovados(Elemento* lst, char turma);
```

Dica: No item B, utilize a função desenvolvida no item A.

INF 1007 – P3 – 23/06/10	
Nome:	
Matrícula:	Turma:

Solução da Questão 1:

```
Elemento* lst_insere_aluno(Elemento* lst, Aluno* novo)
{
   Elemento *p, *n, *ant = NULL;

   for(p=lst; p!=NULL; p = p->prox)
   {
      if(strcmp(p->info->nome, novo->nome) > 0)
         break;
      ant = p;
   }

   n = (Elemento*) malloc (sizeof(Elemento));
   n->info = novo;

   if(ant == NULL)
   {
      n->prox = lst;
      return n;
   }

   n->prox = ant->prox;
   ant->prox = n;

   return lst;
}


Elemento* lst_aprovados(Elemento* lst, char turma)
{
   Elemento *p, *nlst = NULL;
   float media;

   for (p = lst; p != NULL; p = p->prox)
   {
      if (p->info->turma == turma)
      {
         media = (p->info->p1 + p->info->p2 + p->info->p3) / 3;

         if (media >= 5)
            nlst = lst_insere_aluno (nlst, p->info);
      }
   }
   return nlst;
}
```

INF 1007 – P3 – 23/06/10	
Nome:	
Matrícula:	Turma:

Questão 2

[Valor: 3,0 pontos] Considere listas encadeadas em que cada elemento é do tipo estruturado *Elemento*, definido a seguir:

```
struct elemento {
   int info;
   struct elemento *prox;
};
typedef struct elemento Elemento;
```

Considere ainda uma TAD *Pilha* que armazena valores inteiros, cuja interface, definida em um arquivo *pilha.h*, dispõe das funções descritas na tabela abaixo.

Pilha* pilha_cria (void);	Retorna o ponteiro para uma nova pilha alocada dinamicamente.
int pilha_pop (Pilha* p);	Retira um elemento do topo de uma pilha. O ponteiro da pilha é passado como parâmetro, e o retorno é o valor deste elemento.
void pilha_push (Pilha* p, int x);	Insere um elemento no topo de uma pilha. O ponteiro da pilha e o elemento a ser inserido são passados como parâmetros.
int pilha_vazia (Pilha* p);	Verifica se a pilha está vazia. O ponteiro da pilha é passado como parâmetro e o retorno é 1, se a pilha estiver vazia, ou 0, caso contrário.
void pilha_libera (Pilha* p);	Esvazia e libera a memória alocada para uma pilha. O ponteiro da pilha é passado como parâmetro.

Utilizando obrigatoriamente uma *Pilha* como estrutura temporária auxiliar, escreva uma função para verificar se uma lista é o inverso da outra, isto é, se uma lista contêm os mesmos valores inteiros de outra lista, mas encontrados na ordem contrária quando percorremos as listas do início até o final. Os parâmetros da função são os ponteiros ***L1*** e ***L2*** para as duas listas. O retorno da função deve ser 1 se uma lista for o inverso da outra, ou 0, caso contrário. O protótipo é:

```
int Compara_listas(Elemento* L1, Elemento* L2);
```

INF 1007 – P3 – 23/06/10	
Nome:	
Matrícula:	Turma:

Solução da Questão 2:

```
int Compara_listas(Elemento* r1, Elemento* r2)
{
   Pilha* p = pilha_cria();
   Elemento* f;
   int x;

   for (f = r1; f != NULL; f = f->prox) {
      pilha_push(p, f->info);
   }

   for (f = r2; f != NULL && !pilha_vazia(p); f = f->prox)
   {
      x = pilha_pop(p);
      if (x != f->info)
      {
         pilha_libera(p);
         return 0;
      }
   }

   if (f == NULL && pilha_vazia(p))
   {
      pilha_libera(p);
      return 1;
   }
   else
   {
      pilha_libera(p);
      return 0;
   }
}
```

INF 1007 – P3 – 23/06/10	
Nome:	
Matrícula:	Turma:

Questão 3

[Valor: 3,0 pontos] Considere uma *árvore binária de busca* que armazena ponteiros para variáveis do tipo *Aluno*, definido a seguir:

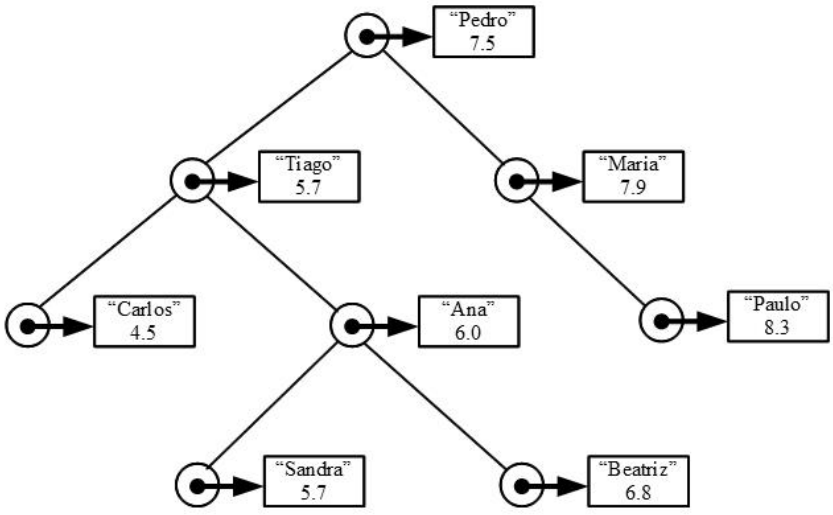
```
struct aluno {
   char nome[81];
   float media;
};
typedef struct aluno Aluno;
```

A árvore está ordenada de tal forma que as médias dos alunos associados aos nós da subárvore à esquerda são menores que a média do aluno associado à raiz e as médias dos alunos associados aos nós da subárvore à direita são maiores ou iguais à média do aluno associado à raiz. O tipo que representa um nó da árvore é dado por:

```
struct noArv {
   Aluno *info;
   struct noArv *esq, *dir;
};
typedef struct noArv NoArv;
```

Escreva uma função que recebe como parâmetros o ponteiro para a *árvore binária de busca* descrita e a média mínima de aprovação, *a* e *m*, respectivamente, e imprime o nome e a média dos alunos aprovados (com média ≥ *m*) em ordem DECRESCENTE. A função deve tirar proveito da ordenação da árvore e obedecer ao seguinte protótipo:

```
void imprime_aprovados(NoArv* a, float m);
```



Por exemplo, considerando a árvore acima, para *m* = 6.5, os nós associados aos alunos "Carlos" e "Sandra" não seriam visitados e a saída seria:

```
Paulo 8.3
Maria 7.9
Pedro 7.5
Beatriz 6.8
```

INF 1007 – P3 – 23/06/10	
Nome:	
Matrícula:	Turma:

Solução da Questão 3:

```
void imprime_aprovados(NoArv* a, float m)
{
   if (a == NULL)
      return;

   imprime_aprovados(a->dir, m);

   if (a->info->media >= m)
   {
      printf("%s %.1f\n", a->info->nome, a->info->media);
      imprime_aprovados(a->esq, m);
   }
}
```

Protótipos de funções que podem ser úteis:

stdio.h:

```
int scanf (char* formato, ...);
int printf (char* formato, ...);
FILE* fopen (char* nome, char* modo);
int fclose (FILE* fp);
int fscanf (FILE* fp, char* formato, ...);
int fprintf (FILE* fp, char* formato, ...);
char* fgets(char* str, int size, FILE* fp));
int sscanf(char* str, char* formato, ...);
```

math.h:

```
double sqrt (double x);
double pow (double x, double exp);
double cos (double radianos);
double sin (double radianos);
```

string.h:

```
int strlen (char* s);
int strcmp (char* s, char *t);
char* strcpy (char* destino, char* fonte);
char* strcat (char* destino, char* fonte);
```

stdlib.h:

```
void* malloc (int nbytes);
void free (void* p);
```

Não separe as folhas deste caderno.