Departamento de Informática - PUC-Rio
INF 1007 – Programação 2
P3 – 26/11/2010

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	–	3,0	
2	–	3,0	
3	–	4,0	
Total		10,0	

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, ...);
```

stdlib.h:

```
void* malloc (int nbytes);
void free (void* p);
void qsort (void *vet, int n, int tam, int (*comp) (const void*,
```

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);
```

Departamento de Informática - PUC-Rio
INF 1007 – Programação 2
P3 – 26/11/2010

Questão 1

[Valor: 3,0 pontos] Seja uma lista encadeada, organizada da seguinte maneira:

1. Um *header* (tipo **Lista**), que armazena a quantidade de elementos da lista (variável **qtdElem**), um ponteiro para o primeiro elemento (variável **prim**) e um ponteiro para o último elemento da lista (variável **ult**). Caso a lista tenha apenas um elemento, ambas as variáveis (**prim** e **ult**) irão apontar para ele;
2. Zero ou mais nós (tipo **Elemento**), onde cada nó armazena a informação em si (variável **info**) e um ponteiro para o próximo nó da lista.

Os tipos de dados envolvidos na manipulação da lista acima são os seguintes:

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

struct lista
{
      int qtdElem;
      Elemento *prim;
      Elemento *ult;
};
typedef struct lista Lista;
```

A função abaixo apresenta o procedimento de criação de uma lista como a descrita anteriormente:

```
Lista* lst_cria(void)
{
   Lista* novo=(Lista*) malloc(sizeof(Lista));
   if(novo==NULL)
   {
         printf("Erro na alocacao de memoria\n");
         exit(1);
   }
   novo->prim=NULL;
   novo->ult=NULL;
   novo->qtdElem=0;
   return novo;
}
```

Você deverá fazer o seguinte: escreva uma função que insira um novo elemento **no final** de uma lista como a descrita acima. Essa função deve obedecer ao seguinte protótipo:

```
void lst_insere_final(Lista* lst,int i);
```

Não separe as folhas deste caderno.

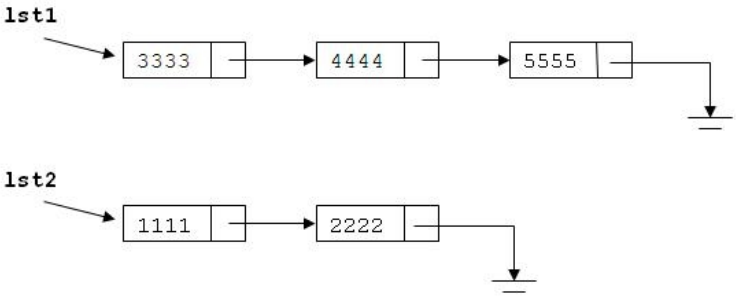
Solução da Questão 1:

Questão 2

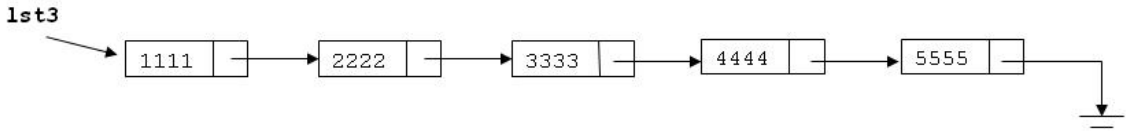
[Valor: 3,0 pontos] Escreva uma função que receba como parâmetros duas listas encadeadas, **lst1** e **lst2**, e usando uma pilha, retorne uma terceira lista, que seja o resultado da concatenação de **lst2** com **lst1**.

A figura a seguir explica o objetivo da questão:

Os nós das listas envolvidas no problema são definidos pela estrutura a seguir:



Resultado da Concatenação




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

Essa função deve obedecer ao seguinte protótipo:

```
Elemento* lst_concatena(Elemento* lst1,Elemento* lst2);
```

Considere que as seguintes funções dos tipos abstratos de dados Pilha e Lista estejam disponíveis:

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.

Não separe as folhas deste caderno.

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.

Elemento* lst_cria(void);	Retorna o ponteiro para uma nova lista alocada dinamicamente.
Elemento* lst_insere(Elemento* lst, int i);	Insere um novo elemento no início de uma lista. O ponteiro para a lista e o elemento a ser inserido são passados como parâmetros.
int lst_vazia(Elemento* lst);	Retorna 1 se a lista estiver vazia, ou 0, caso contrario.
int lst_libera(Elemento* lst);	Libera toda a memória alocada para uma lista. O ponteiro para a lista é passado como parâmetro.

Não separe as folhas deste caderno.

Solução da Questão 2:

Questão 3

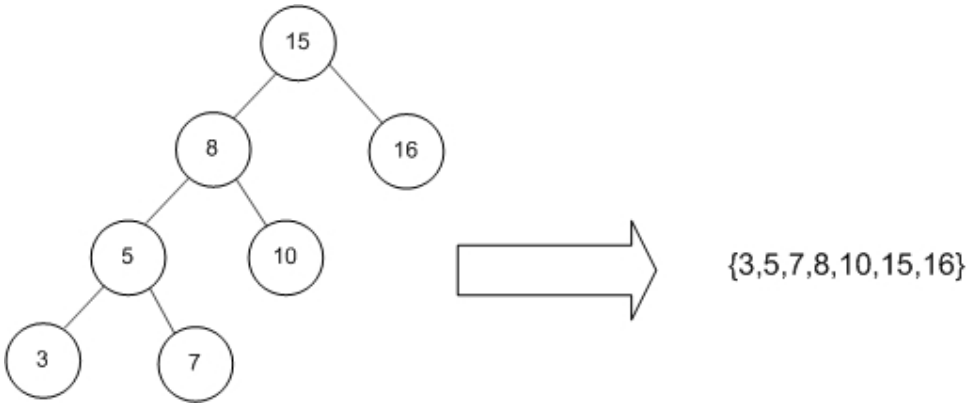
[Valor: 4,0 pontos] Considere uma *árvore binária de busca* contendo números inteiros. Os nós dessa árvore são definidos pela seguinte estrutura:

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

Escreva uma função que imprima, em ordem crescente, os inteiros armazenados na árvore; entre chaves e separados por vírgulas. A função deve obedecer ao seguinte protótipo:

```
void mostra(NoArv *a);
```

A figura a seguir mostra o que deverá ser exibido pela sua função a partir de um exemplo de árvore



Dicas:

- Use uma função auxiliar
- Cuidado para não imprimir uma vírgula após o último número.

Não separe as folhas deste caderno.

Solução da Questão 3: