UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
I.C.M.C.

SCC202 Algoritmos e Estruturas de Dados I – Turma Mat.Aplicada
Profa. Graça Nunes
2º. Semestre de 2010

Prova 1 (Gabarito)
17/09/2009

Nome:_____Nro USP: _____

1) Considere uma lista linear sequencial, implementada como abaixo:

(a) **(1.0)** Escreva um **algoritmo** para verificar se uma lista de inteiros está ordenada crescentemente.

Resp.:

início
A partir do primeiro elemento da lista, seja ai (i=1) o elemento atual;
Enquanto ai for menor ou igual ao seu sucessor ai+1, siga em frente.
Se chegou ao final da lista, retorne TRUE
Se parou porque ai > ai+1, retorne FALSE.
fim

(b) **(1.5)** Considere agora que a lista está representada conforme a declaração abaixo:

```
typedef struct{
   int nelem;  /*número atual de elementos*/
   tipo_elem A[MAX+1];
}Lista;
```

Escreva uma função em C que implemente o algoritmo de (a).

Resp.:

```
Boolean  eh_ordenada(Lista *L) {
int i;
for (i=0; i<= L->nelem; i++) {
  if (L->A[i] > L->A[i+1]) return FALSE; }
return TRUE;
}
```

(c) **(0.5)** Qual é o custo estimado de tempo de sua função, em relação ao tamanho da lista?

É linear. Se a lista tem tamanho n, então o custo é exatamente n, pois é necessário passar por todo elemento da lista, comparando-o com seu sucessor/predecessor.

2) **(1.5)** Considere o TAD Fila Sequencial. Utilize suas funções para implementar uma função que gerencia o empréstimo de um livro numa biblioteca:

```
typedef int usuário;
typedef struct {
char título;
usuário fila_livro[MAX];
} Livro;
```

boolean empresta-livro (Livro *b, usuário *u)

*/*essa função deve retornar **true** se o livro b puder ser emprestado para o usuário u; ou **false**, e neste caso, o usuário deve ficar na fila de espera do livro. Se não for possível inserir na fila, retorne **false***/*

```
{ if (Disponível(b) && VAZIA(b->fila_livro)) return TRUE;
    else {Insere (b->fila_livro, *u); /* supondo que Insere é uma função "void" */
        return FALSE; }
   }
```

3) **(1.0)** Considerando que nas **pilhas** todas as operações ocorrem apenas na extremidade do topo:

(a) Qual é a única desvantagem de se implementar pilhas sequencialmente?

Ter que prever o tamanho do array.

(b) Qual é o único custo de se trocar a implementação sequencial pela encadeada no caso da pilha?

O campo extra para endereço do sucessor de cada elemento.

4) Explique por que a Busca Binária:

(a) **(1.0)** É mais eficiente do que a busca seqüencial.

Em cada passo da busca seqüencial, ao comparar a chave de busca com a próxima chave do conjunto, diminuímos o tamanho do problema em uma unidade. Na busca binária, a cada passo, diminuímos pela metade o tamanho do problema. Assim, no pior caso, quando a

chave de busca não está na lista, a busca seqüencial pode levar até n comparações (n é o tamanho da lista, enquanto a busca binária levaria $\log_2$ n comparações.

(b) **(1.0)** Só pode ser efetuada num array.

> Num array ordenado, temos acesso direto à sua mediana por meio dos seus índices. Ao somar os índices das extremidades e dividir por dois, obtemos o endereço do valor mediano. Isso só é possível quando (a) temos acesso a eles e (b) eles são consecutivos.

5) **(1.0)** Responda cada uma das questões abaixo com V(erdadeiro) ou F(also). No caso de F, corrija ou justifique.

(V) As duas únicas vantagens de um mapeamento encadeado dinâmico de uma lista, sobre o estático, são: (a) evitar deslocamentos dos elementos do array durante inserções e eliminações, e (b) evitar a reserva de memória em tempo de compilação.

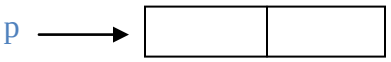
(F) A implementação de uma pilha em formato circular é uma solução para o melhor aproveitamento do espaço em memória quando se utiliza array para armazená-la.
Isso é verdadeiro para Filas, e não para Pilhas, pois nessas últimas, não há necessidade de reaproveitamento de espaço, já que as operações ocorrem numa única extremidade, o topo.

(V) A única razão para se implementar uma lista encadeada de forma estática (num array) é para o caso de se usar linguagens de programação que não admitem variáveis dinâmicas.

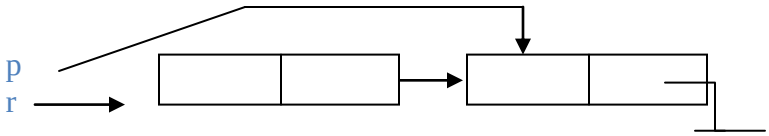
(V) O custo de se encontrar uma chave numa lista encadeada é necessariamente, no pior caso, linear com o tamanho da lista.

6) **(1.5)** Esquematize, com desenhos, o que acontece na memória principal durante a execução do código a seguir:

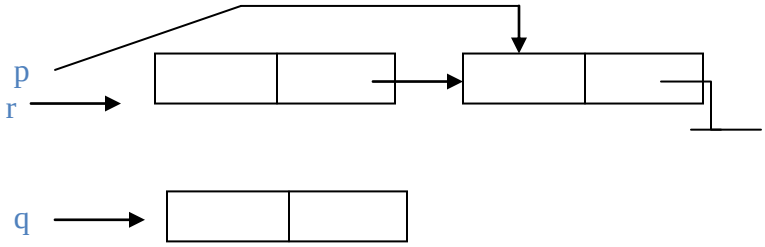
```
struct no {
        char info;
        struct no *next;
};

struct no *p, *q, *r;
p = (struct no*) malloc(sizeof(struct no));
```




```
p->next = (struct no*) malloc(sizeof(struct no));
p->next->next = NULL;
r = p;
p = p->next;
```



q = (struct no*) malloc(sizeof(struct no));



r->next = q;
q->next = NULL;
free(p);
p= NULL;