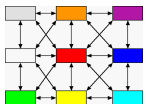
Estruturas de Dados
2015/16
Frequência F1



Universidade da Beira Interior
Departamento de Informática

1. Considere as seguintes estruturas de dados, relativas ao sistema de informação utilizado pela NASA para registar informação sobre as estrelas avistadas num dos seus observatórios astronómicos. Para cada estrela, é necessário guardar informação sobre o observador, a data de observação, a estimativa da distancia média entre a estrela e a Terra (medida em anos-luz) e o nome atribuído pelo observador à estrela.

```
typedef struct OBSERVADOR{
        int codigo;      //critério de ordenação na lista
        char nome[80];
        struct OBSERVADOR *nseg;
} Observador;

typedef struct OBSERVACAO{
       int dia, mes, ano;     /data da observação
       char nome[80];      //critério de ordenação na lista
       Observador *observador;
       float distancia;
       struct OBSERVACAO *nseg[4];      //lista de salto
       int nivel;     /nível na lista de salto
}Observacao;
```

1a) (3 valores) Mediante as estruturas apresentadas, esquematize a organização da informação resultante do armazenamento dos seguintes itens:

- Observador 3, José Andrade;
- Observador 2, Vitor Coelho;
- Observador 4, Andreia Soares;
- Observador 1, Paula Correia

- Observacao 12/10/2014, “xyz14”, Observador 1, 300 anos-luz, nó de nível 0
- Observacao 12/11/2014, “alfa14”, Observador 2, 200 anos-luz, nó de nível 1
- Observacao 2/12/2014, “beta10”, Observador 2, 100 anos-luz, nó de nível 0
- Observacao 12/1/2015, “zz23”, Observador 1, 10 anos-luz, nó de nível 2
- Observacao 10/2/2015, “omega0”, Observador 1, 14 anos-luz, nó de nível 3
- Observacao 15/10/2015, “omega1”, Observador 4, 25 anos-luz, nó de nível 1

1b) (3 valores) Implemente uma função que imprima o nome dos observadores que nunca observaram nenhuma estrela.

Protótipo: void mostraObservadoresNulos(Observador *L, Observacao *LO);

1c) (3 valores) Codifique a função que apaga do sistema um observador e as respectivas observações. A função deverá retornar um apontador para a lista de observadores, enquanto a alteração da lista de observações se repercutirá fora da função através do duplo apontador “Observacao**”.

Protótipo: Observador* apagaObservadorObservacoes(Observador *L, Observacao **LO, int codigoObservador);

Considere que tem à sua disposição a seguinte função:

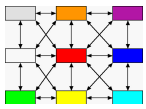
```
Observador* findNode(Observador *L, int codigo);
//Procura por um observador e retorna o apontador para o respectivo elemento (NULL caso não exista na lista).
```

1d) (3 valores) Implemente uma função que conte o número de estrelas observadas num determinado período de tempo, isto é entre “dia1-mês1-ano1” e “dia2-mês2-ano2”.

Protótipo: int totalObservacoes(Observacao *LO, int dia1, int mes1, int ano1, int dia2, int mes2, int ano2);

1e) (3 valores) Implemente uma função que conte (de uma forma eficiente) o número de nós na lista de salto com um determinado nível.

Protótipo: int totalNosNivel(Observacao *LO, int nivel);



2) Considere a seguinte estrutura de dados, relativa à implementação de uma fila de espera para os clientes de um consultório médico.

```
typedef struct QUEUE{
        int hora, minuto, segundo;   //momento de entrada na fila
        char  nomeCliente[80];
        struct QUEUE *nseg;
} Queue;
```

2a) (3 valores) Implemente uma função que retire da fila os elementos há mais que "n" segundos à espera de ser atendidos, e os transfira para a fila "Atrasados". A função deverá retornar um apontador para a cabeça da nova fila "Atrasados".

Protótipo: Queue* transfereAtrasados(Queue **F, int n);

Considere que tem à sua disposição as seguintes funções:

Queue* push(Queue *F, Queue *nv);
//insere um novo elemento numa fila

Queue* pop(Queue **F);
//retira um elemento da fila

int totalSegundos(int hora1, int minuto1, int segundo1, int hora2, int minuto2, int segundo2); //devolve o total de segundos entre "hora1:minuto1:segundo1" e "hora2:minuto2:segundo2".

Queue* makeNode(int h, int m, int s, char *nome);
//Cria um novo nó

2b) (2 valores) Qual o output do seguinte programa?

```
void funcaoX(Queue *F){
   Queue *aux;
   while(F!=NULL){
      aux=pop(&F);
   }
}
void main(){
   Queue *Q, *aux;
   Q=push(NULL, makeNode(0,0,0,"a"));
   Q=push(Q, makeNode(0,1,0,"b"));
   Q=push(Q, makeNode(0,2,0,"c"));
   Q=push(Q, makeNode(0,3,0,"d"));
   funcaoX(Q);
   while (Q!=NULL){
      aux=pop(&Q);
      printf("%s\n",aux->nome);
   }
}
```