

Programação Imperativa

Exame de Recurso

1º Ano – LEI/LCC

8 de Julho de 2015

Parte A

Considere as seguintes definições de tipos:

```
typedef struct slist {  
    int valor;  
    struct slist *prox;  
} *LInt;
```

1. Defina uma função `int bitsUm (unsigned int n)` que calcula o número de bits iguais a 1 usados na representação binária de um dado número `n`.
2. Defina uma função `int limpaEspacos (char t[])` que elimina repetições sucessivas de espaços por um único espaço. A função deve retornar o comprimento da string resultante.
3. Defina uma função `int dumpL (LInt l, int v[], int N)` que, dada uma lista `l`, preenche o array `v` com os elementos da lista. A função deverá preencher no máximo `N` elementos e retornar o número de elementos preenchidos.
4. Defina uma função `LInt parte (LInt l)` que parte uma lista `l` em duas: na lista `l` ficam apenas os elementos das posições ímpares; na lista resultante ficam os restantes elementos.

Assim, se a lista `x` tiver os elementos {12,22,32,42,52,62} a chamada `y = parte (x)`, coloca na lista `y` os elementos {22,42,62} ficando em `x` apenas os elementos {12,32,52}.

Parte B

Considere a seguinte definição para armazenar a informação sobre os alunos de uma turma numa árvore binária de procura ordenada pelo **número** do aluno:

```
typedef struct nodo {
    char nome [50];
    int numero;
    int nota; // >=0, <=20
    struct nodo *esq, *dir;
} *Alunos;
```

1. Defina uma função `int fnotas (Alunos a, int notas [21])` que preenche o *array* `notas` com a frequência das notas da turma (por exemplo, em `notas[12]` deve ser colocado o número de alunos que tiveram nota 12). A função deverá retornar o número de alunos da turma.
2. Defina uma função `int rankAluno (Alunos a, int numero, int fnotas[21])` que, dada uma turma, a frequência de notas (tal como calculado na alínea anterior) e o número de um aluno da turma, calcula o *rank* de um dado aluno numa turma. Um aluno tem *rank* N sse existirem N-1 alunos com nota superior.

Sugestão: Comece por definir uma função `int rankNota (int nota, int fnotas[21])` que, dada uma nota e a frequência de notas calcula o *rank* dessa nota.

3. Considere o seguinte tipo para representar uma lista de *strings*.

```
typedef struct strlist {
    char *string;
    struct strlist *prox;
} *StrList;
```

Defina uma função `int comNota (Alunos a, int nota, StrList *l)` que coloca em `*l` a lista dos nomes dos alunos da turma `a` que tiveram essa nota. A função deve retornar o número de alunos nessas condições (i.e., o comprimento da lista produzida).

4. Considere agora a seguinte função que se pretende que imprima os nomes dos vários alunos por ordem decrescente da sua nota:

```
void imprime (Alunos a) {
    int i;
    int notas [21];
    int quantos = fnotas (a, notas);
    Alunos todos [quantos];

    preenche (a, todos, notas);

    for (i=0; i<quantos; i++)
        printf ("%d_%s_%d\n",
                todos[i]->numero,
                todos[i]->nome,
                todos[i]->nota);
}
```

Apresente uma definição da função `void preenche (Alunos a, Alunos t[], int freq[21])` de forma a que a função acima imprima por ordem decrescente da nota a informação sobre os alunos guardada na árvore.

Note que o array deve ser preenchido com os endereços dos vários nodos da árvore sem que seja alocada qualquer memória adicional.

Se possível, use a informação presente no array `freq` para otimizar a função.