MC 102Z – Terceira prova - 22/11/12

1) Considere a seguinte função

```
int f(int x, int y){
    if((x == 0) && (y == 0)) return 0;
    if(x == 0) return 1 + f(0, y / 2);
    if(y == 0) return 1 + f(x / 2, 0);
    return 2 + f(x/2,y/2);
}
```

a) Quais os valores retornados para as seguintes chamadas a **f()** ? (valor 1.0)

```
f(15,0);
f(0,15);
f(6,14);
f(14,14);
```

```
f(15,0) = 1 + f(7,0) = 1 + 1 + f(3,0)
        = 1 + 1 + 1 + f(1, 0) = 1 + 1 + 1 + 1 + f(0,0) = 4
f(0,15) = 1 + f(0, 7) = 1 + 1 + f(0, 3)
        = 1 + 1 + 1 + f(0, 1) = 1 + 1 + 1 + 1 + f(0,0) = 4
f(6,14) = 2 + f(3,7) = 2 + 2 + f(1,3) =
        = 2 + 2 + 2 + f(0,1) = 2 + 2 + 2 + 1 + f(0,0) = 7
f(14,14)= 2 + f(7,7) = 2 + 2 + f(3,3) = 2 + 2 + 2 + f(1,1)
        = 2 + 2 + 2 + 2 + f(0,0) = 8
```

b) Escreva uma função não recursiva equivalente a **f()** . (valor 2.0)

R: a função divide cada parâmetro por 2 até que os dois sejam iguais a zero e retorna o número de divisões realizadas.

```
int f(int x,int y){
      int i = 0;
      for( ; x != 0; i++) x /= 2;
      for( ; y != 0; i++) y /= 2;
      return i;
}
```

2) A execução do programa "**exemplo.c**" gerou a saída mostrada abaixo (notar que a 'saída' corresponde a todo o texto do quadro). Explique o que acontece. (valor 2.0)

```
 1  #include <stdio.h>
 2  #define LINE_LENGTH 80
 3
 4  int nlinhas(char *nome){
 5    FILE* fp;
 6    char line[LINE_LENGTH];
 7    int count=0;
 8    fp=fopen(nome,"r");
 9    while ( fgets(line, LINE_LENGTH, fp) != NULL) {
10        printf("%5d\t%s",++count,line);
11    }
12    fclose(fp);
13    return count;
14  }
15
16  int main(){
17     printf("\n *** no. de linhas: %d **** \n",nlinhas("exemplo.c"));
18  }

*** no. De linhas: 18 ****
```

R: A função nlinhas le cada linha de um arquivo texto e imprime, numerando cada linha. Essa função tem como parâmetro o nome do arquivo a ser lido e retorna o número de linhas lidas. A função main chama essa função e escreve o número de linhas lidas. No programa apresentado, o arquivo texto lido por nlinhas é o próprio programa 'exemplo.c'.

3) Considere as definições a seguir.

```
struct aluno {
              int ra;
              char *nome;
              int curso;
       };
```

```
struct aluno *novoAluno(int ra, char* nome, int curso){
     struct aluno* p = malloc(sizeof(struct aluno));
     (*p).ra = ra;
     (*p).nome = malloc(strlen(nome)+1);
     strcpy((*p).nome,nome);
     (*p).curso = curso;
     return p;
}
```

a) Escreva a função para liberar uma estrutura do tipo 'aluno', alocada através da função **novoAluno()** (valor 1.0). Sua função deve obedecer ao seguinte protótipo:

```
void liberaAluno(struct aluno *p);
```

```
void liberaAluno(struct aluno *p){
     free((*p).nome);
     free(p);
}
```

b) Escreva uma função para escrever uma turma de alunos num arquivo (valor 2.0). Sua função deve obedecer ao seguinte protótipo:

```
void escreveTurma(struct aluno* turma[], int nAlunos, char* nomeDoArquivo);
```

```
void escreveTurma(struct aluno* turma[], int nAlunos, char* nomeDoArquivo){
      FILE* arq = fopen(nomeDoArquivo,"w");
      int i;
      for(i = 0; i < nAlunos; i++)
          fprintf(arq,"%d %d %s\n",(*turma[i]).ra,(*turma[i]).curso,(*turma[i]).nome);
      fclose(arq);
}
```

c) Escreva uma função para ler os dados de uma turma de um arquivo criado através da função definida no item anterior (valor 2.0). Sua função deve obedecer ao seguinte protótipo:

```
void leTurma(struct aluno *turma[], int nAlunos, char* nomeDoArquivo);
```

```
void leTurma(struct aluno* turma[], int nAlunos, char* nomeDoArquivo){
      FILE* arq = fopen(nomeDoArquivo,"r");
      char nome[80]; int ra, int curso;
      int i;
      for(i = 0; i < nAlunos; i++)
          fscanf(arq,"%d %d %s\n",ra, curso, nome);
          turma[i] = novoAluno(ra,nome,curso);
      fclose(arq);
}
```