PR – LP1 - 2013-2
Prof. Leandro Marzulo

Nome:

Instruções: Esta prova é composta de duas questões totalizando 12 (doze) pontos, sendo a nota máxima 10 (dez). Responda as questões de forma sucinta e clara e usando indentação para as questões que envolvem programação. O uso de lápis é permitido, entretanto, pedidos de revisão serão considerados apenas para questões respondidas a caneta. BOA PROVA!

Questão 1) (2,0) Explique as diferenças entre struct e union, quanto a função, declaração e forma de uso.

Structs são usadas para agrupar variáveis de tipos diferentes. Cada membro da estrutura (acessado com o operador . ou ->) é uma variável separada com seu próprio espaço de armazenamento. Unions são usadas quando a informação que precisamos armazenas pode ter tipo variado. Cada membro de uma union (acessado também com o operador . ou ->) indica de que maneira a informação armazenada na union será interpretada (de acordo com o tipo do membro). A union então contem apenas um espaço de armazenamento do tamanho do maior membro e só pode armazenas um membro por vez

A declaração é bem semelhante:

```
struct teste
{
        int a;
        double n;
        char c;
} myteste;
```

Na struct myteste podemos armazenar simultaneamente um int, um double e um char.

```
union teste
{
        int a;
        double n;
        char c;
} myteste;
```

Na union myteste podemos armazenar ou um int, ou um double, ou um char.

Questão 2) (10,0) Considere um projeto de uma aplicação que leia um vetor V de números inteiros e um outro inteiro X e imprima, ao final o valor de V+X.

Este projeto de software está dividido em 3 arquivos:

- principal.c – arquivo que contém a implementação da função main. Esta função faz a declaração de V, faz a leitura de X do teclado e chama as funções leVetor, somaVetorEscalar e imprimeVetor. V é alocado estaticamente e o tamanho de V é definido em uma constante neste mesmo arquivo
- auxiliar.c – arquivo que contém a implementação das seguintes funções
    - leVetor – recebe como parâmetro o ponteiro para o vetor e o seu tamanho e lê do teclado todos os seus elementos.

- somaVetorEscalar - que recebe como parâmetros de entrada um vetor de inteiros V, o tamanho do vetor e um escalar E e executa a operação V=V+E usando apenas aritmética de ponteiros para acessar os elementos de V.
- imprimeVetor - recebe como parâmetro o ponteiro para o vetor e o seu tamanho e imprime na tela todos os elementos de V, separados por quebras de linha.

- auxiliar.h – arquivo que contém o protótipo das funções declaradas no arquivo auxiliar.h

a) (1,0) Escreva o código contido no arquivo principal.c.

```
#include <stdio.h>
#include <auxiliar.h>

#define TAM 10

int main(void)
{
        int V[TAM];
        int x;
        printf("Digite o escalar: ");
        scanf("%d",&x);
        getchar(); //para consumir a quebra de linha
        leVetor(V,TAM);
        somaVetorEscalar(V,TAM,x);
        imprimeVetor(V,TAM);
        return 0;
}
```

b) (3,0) Escreva o código contido no arquivo auxiliar.c

```
#include <stdio.h>
#include <auxiliar.h>

void leVetor(int * V, int tam)
{
        int i;
        for (i=0;i<tam;i++)
        {
                printf("V[%d]: ", i);
                scanf("%d", &V[i]);
                getchar(); //consumindo a quebra de linha
        }
}

void somaVetorEscalar(int * V, int tam, int e)
{
        int i;
        for (i=0;i<tam;i++)
                (*(V+i))+=e;
}

void imprimeVetor(int * V, int tam)
{
        int i;
        printf("Resultado\n");
        for (i=0;i<tam;i++)
                printf("%d\n",V[i]);
}
```

c) (1,0) Escreva o código contido no arquivo auxiliar.h

```
void leVetor(int *, int);
void somaVetorEscalar(int *, int, int);
void imprimeVetor(int *, int);
```

d) (1,0) Reescreva o código contido no arquivo principal.c, considerando que V é alocado dinamicamente e o tamanho de V é lido do teclado.

```
#include <stdio.h>
#include <stdlib.h>
#include <auxiliar.h>

int main(void)
{
        int *V;
        int x;
        int tam;

        printf("Digite o escalar: ");
        scanf("%d",&x);
        getchar(); //para consumir a quebra de linha

        printf("Digite o tamanho do vetor: ");
        scanf("%d",&tam);
        getchar(); //para consumir a quebra de linha

        V = (int *) malloc(tam*sizeof(int));

        leVetor(V,tam);
        somaVetorEscalar(V,tam,x);
        imprimeVetor(V,tam);
        return 0;
}
```

e) (2,0) Reescreva o código da função leVetor para que ela receba, além dos demais parâmetros, uma string contendo o nome de um arquivo. A função deve abrir este arquivo e ler os elementos do vetor deste arquivo ao invés de ler do teclado.

```
void leVetor(int * V, int tam, char * fName)
{
        FILE * fp;
        if (!(fp = fopen(fName, "r")))
        {
                printf("Erro ao abrir o arquivo %s!\n", fName);
                exit(1);
        }
        int i;
        for (i=0;i<tam;i++)
                if (fscanf(fp,"%d\n", &V[i])<=0)
                {
                        printf("Erro ao ler o arquivo %s!\n", fName);
                        exit(1);
                }
        fclose(fp);
}
```

f) (2,0) Reescreva o código da função somaVetorEscalar para fazer o calculo V=V+E de forma recursiva.

```
void somaVetorEscalar(int * V, int tam, int e)
{
        if (tam)
        {
                (*V)+=e;
                somaVetorEscalar(V+1, tam-1, e);
        }
}
```