MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO

Engenharia de Computação – Campus Nova Gameleira
Disciplinas: Programação de Computadores I (PC I) e Lab. PC I– Turmas Extras - Semestre: 2º/2022
Prof.: Evandrino Barros (evandrino@cefetmg.br)

Lista 2 – Funções, Vetores e *Strings* - em duplas - 5 pontos – entrega somente pelo AVA.

Gerar um arquivo PDF com os códigos fontes em C dos programas solicitados abaixo, incluindo as telas de execução de cada um. Utilize os recursos do seu computador para imprimir o resultado das telas. Todos os recursos da Linguagem C para solução desse exercício são apresentados nas transparências das Aula 1 a 9 de PC I, presentes no AVA da disciplina.

A entrega deve ser feita pelo AVA da disciplina no link disponível para isso. Envie somente um PDF por dupla. Inclua o nome de cada integrante no documento. Inclusões posteriores não serão aceitas.

1. Indique se são verdadeiras ou falsas as seguintes afirmações:

a) Uma função em ***C*** pode devolver simultaneamente mais do que um valor.
b) Uma função em ***C*** pode não ter parâmetros.
c) Uma função em ***C*** tem que devolver sempre um inteiro.
d) Os parâmetros das funções podem ser do tipo ***void***.
e) A instrução ***return*** termina a execução de uma função.
f) Uma variável local a uma função pode ter o mesmo nome que um parâmetro.
g) A instrução ***return*** termina a execução de uma função apenas se for a última instrução da função em que se encontra.
h) A instrução ***return***, quando executada dentro de qualquer função, termina o programa.
i) A instrução ***return***, quando executada dentro da função ***main***, termina o programa.
j) O nome de uma função é opcional.
k) Os parâmetros numa função são opcionais.
l) Uma função deve fazer o maior número de tarefas possível sem ocupar muito código.
m) Uma função não deve ter mais que 10 linhas.
n) O nome de uma função não deve ter mais do que 6 letras.
o) O nome de uma função não pode ser uma palavra reservada do ***C***.
p) Sempre que for necessário devem ser utilizadas variáveis locais.
q) Um protótipo não é nada mais que a repetição do cabeçalho da função seguido de;.
r) Em ***C***, um procedimento não é mais do que uma função que "retorna ***void***"

2) Identifique os erros de compilação quer seriam detectados nos seguintes programas

a)
```
f(int x,int y);
{
  x = 4;
  y = 5;
}
```

b)
```
void f(int x,int y)
{
    return -1;
}
```

c)
```
void f(void);
void f(int x,int y)
{
  x = 4;
  y = 5;
}
```

d)
```
f(int x,int y);
void f(int x,int y)
{
  x = 4;
  y = 5;
}
```

e)
```
void (int x,int y)
{
  x = 4;
  y = 5;
}
```

f)
```
void f(int x, y)
{
  x = 4;
  y = 5;
}
```

3) Implemente as seguintes funções:

a) **int Abs (int x)**

Devolve o valor absoluto de x.

```
Abs (-5)    → 5
Abs (5)     → 5
```

b) **float Pot (float x, int n)**

Devolve o valor de x^n

$x^0 = 1.0$
$x^n = x * x * \ldots * x$ *(n vezes)*

c) **float VAL (float x, int n, float t)**

Devolve o VAL (Valor atual Líquido) para n anos, à taxa t e é definido através da seguinte fórmula

$$VAL = \frac{x}{(1+t)} + \frac{x}{(1+t)^2} + \frac{x}{(1+t)^3} + \ldots + \frac{x}{(1+t)^n}$$

Utilize a função Pot, implementada anteriormente (questão 3.b).

d) **long int num (int *n_horas*, char *tipo*)**

Recebe a quantidade de horas pelo parâmetro *n_horas* e retorna a quantidade de minutos ou segundos, de acordo com o segundo parâmetro (tipo), que tipo pode ser 'm' para minutos e 's' para segundos.

```
num(3, 'm') --> 180
num(3, 's') --> 10800
```

/* Resolva este exercício de 2 formas distintas: com a instrução if-else e com switch com break */

Nota: Supõe-se que o tipo está sempre correto.

Ao final, implemente um programa que chame todas as funções acima a partir da função **main** do programa.

4) Indique quais os erros de programação ou de compilação que os seguintes trechos apresentam:

a)
```
#include <stdio.h>
main()
{
    int v[10],i;
    for (i=1;i<=10;i++)
        v[i] = 0;
}
```

b)
```
#include <stdio.h>
main()
{
    int v[10],i;
    for (i=0;i<10;i++)
        v[i] = 0;
    v[i] = 101;
}
```

c)
```
#include <stdio.h>
main()
{
    int i=5;
    int v[i];
    for (i=0;i<5;i++)
        v[i] = 0;
}
```

d)
```
#include <stdio.h>
main()
{
    int v[],i;
    for (i=0;i<10;i++)
        v[i] = 0;
    v[i] = 101;
}
```

e)
```
#include <stdio.h>
main()
{
    int i;
    int v[3] = {10,20,30,40,50};
    for (i=0;i<10;i++)
        v[i] = 0;
    v[i] = 101;
}
```

f)
```
#include <stdio.h>
#define MAX 30;
main()
{
    int v[MAX];
    for (i=0;i<10;i++)
        v[i] = 0;
    v[i] = 101;
}
```

5) Implemente as seguintes funções que envolvem vetores e matrizes

a) **float max (float v[ ], int n)**

Que recebe um vetor de números reais e o número de elementos a considerar.
Retorna o maior número entre os n primeiros elementos do vetor.

b) **void transpor (int v[MAX][MAX])**

Que transpõe a matriz v com MAX por MAX elementos.

c) **int memicmp (char *s1, char *s2, int n)**

Que verifica se as n primeiras posições dos vetores sl e s2 são ou não iguais, independentemente de estarem em maiúsculas ou minúsculas (ignore case).

Ao final, implemente um programa que chame todas as funções acima a partir da função **main** do programa.

6) Implemente as seguintes funções que envolvem *strings*

a) **int strcounta (char *s)**

Devolve o nº de caracteres alfabéticos em s.

```
strcounta ("15 abacates") → 8
strcounta ("quinze (15) abacates") → 14
```

b) **int stricmp (char *sl, char *s2)**

Faz o mesmo que a função strcmp da biblioteca *string.h*, mas realiza a comparação ignorando se os caracteres estão em maiúsculas ou minúsculas (ignore case).

c) **char *strduplica (char *s)**

Recebe uma string e duplica o seu conteúdo.

```
char str[100] = "Ana";
strduplica(s) > "AnaAna"
```

d) **char *UpDown (char *s)**

Coloca os caracteres da string s alternadamente em Maiúsculas e Minúsculas.

```
char * MyString = "Alfabeto Grego";
UpDown (MyString); → "AlFaBeTo gReGo"
```

e) **int atoi (char *s) /* Array to Integer */**

Recebe uma string e devolve o inteiro que nela está representado.

```
atoi ("1234") → 1234
atoi("-123abc") → -123
atoi ("+51ab46") → 51
atoi ("abc") → 0
```

Novamente, ao final, implemente um programa que chame todas as funções acima a partir da função **main** do programa.