Lista de Exercícios sobre Funções

1) Simule o funcionamento (no papel) do seguinte programa. Quais valores são impressos na saída padrão?

```c
#include <stdio.h>
#include <stdlib.h>
void f1(int a){
   printf ("%d", a);
   a += a;
   printf ("%d", a);
}

int main(int argc, char **argv) {
   int a = 2;
   int b = 3;
   f1 (a);
   f1 (b);
   printf ("%d %d", a, b);
   return 0;
}
```

2) Simule o funcionamento (no papel) do seguinte programa. Quais valores são impressos na saída padrão?

```c
#include <stdio.h>
#include <stdlib.h>
void f2(int a []){
   printf ("%d", a[1]);
   a[0]++;
   printf ("%d", a[0]);
}

int main(int argc, char **argv) {
   int x [] = {3,2};
   f2(x);
   printf ("%d %d", x[0], x[1]);
   return 0;
}
```

3) Simule o funcionamento (no papel) do seguinte programa. Quais valores são impressos na saída padrão?

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

int f3(int a [], int b){
      int i;
      for (i = 0; i < b; i++){
            if (a[i] > 4){
                  a [i] += 2;
            }
            else{
                  a[i] -= 2;
            }
            a[i]++;
      }
      b++;
      return a[0] + a[1];
}

int main() {
      int x [] = {5, 6, 4, 1};
      int b = 4;
      int resultado = f3(x, b);
      printf ("%d %d %d %d %d %d %d", x[0],
x[1], x[2], x[3], x[4], b, resultado);
      return 0;
}
```

4) Escreva uma função que recebe um valor real como parâmetro que representa uma temperatura em graus Fahrenheit e devolve este valor convertido para graus Celsius. Escreva uma outra função que recebe um valor real como parâmetro que representa uma temperatura em graus Celsius e devolve este valor convertido para graus Fahrenheit. Na função main, escreva um loop que permite ao usuário escolher que tipo de conversão deseja fazer.

Este loop termina quando o usuário digitar a opção -1. Caso o usuário digite 1, faça a conversão de Celsius para Fahrenheit usando a função descrita, e mostre o resultado. Caso o usuário digite 2, faça a conversão de Fahrenheit para Celsius usando a função descrita e mostre o resultado. Somente a função main pode fazer chamadas às funções printf e scanf.

As fórmulas para conversão são as seguintes:
`°F = °C × 1,8 + 32`
`°C = (°F − 32) / 1,8`

5) O algoritmo de ordenação conhecido como SelectionSort (ou ordenação por seleção) baseia-se na seguinte estratégia:

Encontre o menor elemento e o coloque na primeira posição, copiando o elemento que estava na primeira posição para a posição em que estava o menor elemento. Encontre o segundo menor elemento e o coloque na segunda posição, copiando o elemento que estava na segunda posição para a posição em que estava o segundo menor elemento. E assim por diante.

Observe que esta ideia difere fundamentalmente da ideia empregada pelo algoritmo de ordenação conhecido como bubbleSort, que foi visto em aula, embora ambos resolvam o mesmo problema.

Utilizando o cabeçalho de função a seguir, implemente o algoritmo de ordenação por seleção. O vetor `a` é o vetor a ser ordenado e o parâmetro `n` representa o número de elementos contidos no vetor. Escreva também uma função main, cliente de `selection_sort`, que testa a função `selection_sort` com um vetor preenchido com valores digitados pelo usuário.

```
void selection_sort(int a [], int n;
```

Observação: É muito fácil colocar o nome selectionSort em algum buscador e encontrar implementações em qualquer linguagem de programação, o que não o levará a nada. A ideia é que você passe algum tempo tentando fazer a sua própria implementação e aprenda de verdade com esta experiência.

6) Escreva uma função com tipo de retorno `float`, chamada `calculadora` e com uma lista de parâmetros composta por um char, e dois inteiros. Sua função deve utilizar uma estrutura switch/case para verificar qual o valor do char, (+, -, * ou /). Como `char` é um tipo específico de inteiro, o switch/case pode ser usado naturalmente. Sua função deve fazer a operação matemática representada pelo símbolo contido no char, usando como operandos os dois inteiros recebidos como parâmetro e devolver o resultado. Cuidado com a divisão por zero. Escreva uma função `main` para testar a função calculadora com valores digitados pelo usuário.

7) No cabeçalho de função `<stdlib.h>` existe uma função chamada `rand`. Esta função permite a geração de números pseudoaleatórios entre 0 e RAND_MAX, que é um número inteiro definido dentro do próprio cabeçalho, que para muitas implementações do compilador é 32767. Ou seja, a cada vez que seu programa chamar a função `rand`, que não recebe parâmetro algum, ela devolverá um número entre 0 e 32767. É natural a necessidade de números aleatórios para implementar um jogo, por exemplo. Mas muitas vezes é preciso limitar o intervalo dos números que são gerados. Por exemplo, se você quiser escrever um simulador de lançamento de dados, provavelmente vai querer utilizar números aleatórios entre 1 e 6, para representar cada face de um dado. Com algumas manipulações matemáticas, é possível escolher este intervalo. Considere a seguinte função, ainda sem implementação:

```
int gera_aleatorio (int n);
```

7.1 Implemente a função `gera_aleatorio.` Ela deve devolver um número inteiro aleatório entre 0 e n-1. Dica: Considere o operador módulo.

7.2 Implemente a segunda versão de `gera_aleatorio` abaixo. Ela deve devolver um número inteiro entre primeiro e ultimo.

```
int gera_aleatorio2 (int primeiro, int ultimo);
```

8) Utilizando o conhecimento adquirido na questão 7, resolva o seguinte problema: construir um simulador de lançamento de dados que realiza o lançamento de um dados 6000 vezes. As faces são geradas randomicamente e

variam entre 1 e 6. O simulador deve ser implementado por uma função chamada `lanca_dado`. O resultado do problema é um vetor que contém o número de vezes que cada face apareceu. Construir duas versões diferentes dessa função: uma cujo vetor resultante é parâmetro de saída, outra cujo vetor é devolvido pelo return.

9) Considere o seguinte programa:

```
void f1 (){
      f2();
}
```

```
void f2(){
      f1();
}
```

```
int main() {
      f1();
}
```

Estude o que este programa faz. Sem executá-lo no computador, explique o problema existente.

10) Considere o seguinte problema: dado um número, devolver um vetor com todos os seus divisores. Esse problema deve ser resolvido utilizando-se várias funções. Tente modularizar o máximo o seu programa, tendo em mente o reaproveitamento de código. A função final que devolver o vetor deve ser feita de duas maneiras diferentes: uma com o vetor como parâmetro de saída, outra devolvendo o vetor por return.

Para lembrar: Passar um argumento por referência (ou por endereço) significa que o endereço de memória do argumento é copiado para o parâmetro correspondente, de modo que o parâmetro se torna uma referência (ponteiro) indireta ao argumento real. Exemplo:

```
void troca(int *x, int *y) {
   int aux;
   aux=*x;
   *x=*y;
   *y=aux;
}
```

```
int main() {
   int a=0,b=5;
   troca(&a,&b);
   printf("%d %d",a,b);
   return 0;
}
```

Neste exemplo a passagem de parâmetro é ilustrada na chamada troca (&a,&b) de main. Nesta chamada, os endereços dos argumentos a e b, escritos da forma &a e &b são passados para as variáveis ponteiro x e y. Assim, os valores de a e b são atualizados de acordo com as modificações em x e y.
Os argumentos são passados por referência nos casos em que o valor real do argumento deve mudar durante a vida da chamada e também quando o argumento passado ocupa muita memória, como no caso de uma matriz grande. Nesses casos, a sobrecarga de espaço e tempo necessária para criar uma cópia completa do objeto, se ele fosse passado por valor, são evitados, pela cópia de um único endereço para o parâmetro.

11) Considere o código a seguir usando passagem por referência. Analise o código e explique o resultado mostrando passo a passo as alterações ocorridas no vetor a. Verifique se a execução desse código produz algum efeito prejudicial à legibilidade.

```
void incrementa (int *x, int *y) {
   *x = *x + (*y);
   (*y)++;
}
```

```
int main(){
   int a[] = {1,2,3};
   for (int i=0; i<3; i++){
      incrementa(&a[i],&a[1]);
      printf("\n %d",a[i]);
   }
}
```

3) Escrever um programa que recebe um número inteiro representando a quantidade total de segundos e, usando passagem de parâmetros por referência, converte a quantidade informada de segundos em Horas, Minutos e Segundos. Imprimir o resultado da conversão no formato HH:MM:SS. Utilize o seguinte protótipo:

```
void converteHora (int total_segundos, int *hora, int *min, int *seg);
```