Oficina de Programação em C (ICP037)

Prof. Ronald Souza – IC/UFRJ

Prova de Reposição – 18/07/2024

GABARITO

Questão 1) (1.0 ponto)

Que sequência o programa abaixo imprimirá? *Na sua resposta, basta informar o que será impresso. Não é preciso descrever o passo a passo.*

```
#include <stdio.h>
void sequencia(int x) {
      if (x % 10) {
            printf("%d", x);
            sequencia(x - 1);
            printf("%d", x);
      }
}
int main() {
      sequencia(2);
      return 0;
}
```

RESPOSTA: 2112

Questão 2)

a) **(1.0 ponto)** Implemente a função ***totalLetra,*** que recebe uma **string** e também uma letra (**caractere**) qualquer e retorna um **inteiro** correspondente ao total de vezes que essa letra ocorre na string.

RESPOSTA:

```
int total(char v[], char letra) {
      int t = 0;
      for (int i = 0; v[i] != '\0'; i++)
         if(v[i] == letra)
            t++;
      return t;
}
```

Ou simplesmente:

```
int total(char v[], char letra) {
      int t = 0;
      for (int i = 0; v[i] != '\0'; i++)
          t += v[i] == letra; //Pois uma avaliação lógica produz 0 ou 1.
      return t;
}
```

Também possível, mas não recomendado por ser menos legível - tudo dentro do "for":

```
int total(char v[], char letra) {
      int t = 0;
      for (int i = 0; v[i] != '\0'; t += v[i] == letra, i++);
      return t;
}
```

Ainda outra possibilidade - versão recursiva:

```
int total(char v[], char letra, int pos) {
      return (v[pos] == '\0') ? 0 : (v[pos] == letra) + total(v, letra, pos + 1);
}
```

A chamada **inicial dessa versão recursiva assume pos == 0.*

b) **(1.0 ponto)** A função abaixo recebe um vetor de inteiros de tamanho N. Cada posição do vetor de entrada é **necessariamente** um único dígito entre 0 e 9. **Explique textualmente o que faz a função abaixo**:

```
void funcao(int vet[], int n) {
      int f[10] = { 0 }; //Zera todas as posições do vetor f.
      for (int i = 0; i < n; i++)
            f[vet[i]]++;
      for (int i = 0; i < 10; i++)
            printf("%d: %d\n", i, f[i]);
}
```

RESPOSTA:
Computa a **frequência** (f) de cada dígito em ***vet***, i.e. quantas vezes cada dígito entre 0 e 9 ocorre em ***vet.***

COMENTÁRIO: Considere, por exemplo, que o vetor ***vet*** de entrada tem 15 posições, e que seu conteúdo é

```
vet = {5, 2, 8, 8, 1, 1, 1, 9, 0, 1, 2, 1, 0, 5, 8}
```

Assim, o **status final** do vetor ***f*** para o vetor ***vet*** acima será:

```
f = {2, 5, 2, 0, 0, 2, 0, 0, 3, 1}
```

Note que o dígito 0 ocorreu, ao todo, 2 vezes ao longo do vetor ***vet*** (posição 0 de ***f***); o dígito 1 ocorreu 5 vezes em ***vet*** (posição 1 de ***f***), e assim em diante. O vetor ***f*** também deixa claro que não houve nenhuma ocorrência dos dígitos 3, 4, 6 e 7 dentro de ***vet***.

Refinando a explicação: O vetor ***f*** computa o total de ocorrências de cada dígito decimal em ***vet***, e então possui **necessariamente** tamanho 10 pois há apenas 10 valores possíveis em cada posição de vet: algum dígito entre 0 e 9. Cada índice de ***f***, portanto, corresponde a um dígito. Assim, o vetor ***f*** é criado com todas as suas 10 posições iniciadas em 0.
No trecho abaixo, percorre-se todo o vetor ***vet***, e a cada dígito encontrado, aumenta-se unitariamente o seu respectivo total em ***f***:

```
for (int i = 0; i < n; i++)
      f[vet[i]]++;
```

No vetor ***vet*** do exemplo acima, quando i == 0, temos que vet[i] == vet[0] == 5 e, portanto, f[vet[i]] == f[5]. Assim, f[vet[i]]++ equivale a f[5]++, aumentando de 1 unidade o valor da posição 5 do vetor ***f***:

```
f = {0, 0, 0, 0, 0, 1, 0, 0, 0, 0} //A posição 5 aumentou de 1 unidade.
```

Ao término da varredura de ***vet*** desse exemplo, teremos o status final de ***f*** tal como fornecido mais acima.

Questão 3 (1.0 ponto)

Deseja-se implementar uma função que retorna a **soma dos dígitos** de um número de entrada. Por exemplo, para o número 1021 a função retornará 4, pois 1+0+2+1 = 4.

Foi iniciada a escrita da função, conforme o código abaixo:

```
int somaDigitos(int a) {
    // O que retornar?
}
```

→ No seu caderno de respostas **(e não aqui!)**, indique claramente, **em letra de fôrma**, dentre as alternativas abaixo, a que contém o corpo da função que captura corretamente o comportamento acima descrito:

```
A) return (a) ? a % 10 + somaDigitos(a / 10) : 0;
B) return (a % 10) ? a % 10 + somaDigitos(a / 10) : 0;
C) return (a == 0) ? 0 : somaDigitos(a / 10);
D) return (a / 10) ? a % 10 + somaDigitos(a / 10) : 0;
E) return (a) ? a / 10 + somaDigitos(a % 10) : a;
```

RESPOSTA: Letra **A)** `return (a) ? a % 10 + somaDigitos(a / 10) : 0;`

Questão 4) Trabalharemos com o Tipo Abstrato de Dado (TAD) **`Cor`**. Para isso, siga os seguintes passos:
a) **(1.0 ponto)** Crie o **tipo estruturado `Cor`**. Sua estrutura conterá 3 **inteiros**, ***r***, ***g*** e ***b***, referentes aos **canais** vermelho (*red*), verde (*green*) e azul (*blue*).

RESPOSTA:
```
typedef struct {
      int r, g, b;
} cor;
```

b) **(1.0 ponto)** No sistema RGB, o valor de cada um dos 3 canais deve **necessariamente** estar **entre 0 e 255** (inclusive), permitindo assim representarmos um total de 256 * 256 * 256 = 16.777.216 cores! Crie uma **função** chamada ***nivel,*** que realizará tal defesa. Especificamente, sua função ***nivel*** deve receber **um único inteiro *a* de entrada** e então retornar o **inteiro**:
(i) 0, se ***a*** < 0; (ii) 255, se ***a*** > 255; (iii) o próprio ***a*** para os demais casos (ou seja, quando 0 <= ***a*** <= 255).

RESPOSTA:
```
int nivel(int a) {
      return (a < 0) ? 0 : (a > 255) ? 255 : a;
}
```

c) **(1.5 ponto)** Agora, crie uma função que, dados **3 inteiros**, ***x, y*** e ***z*** **quaisquer** de entrada, retorne uma **`Cor`** cujos canais ***r***, ***g*** e ***b*** valem, respectivamente**,** ***x***, ***y*** e ***z***. **Defenda os valores dos canais**: o valor de todo canal deve estar entre 0 e 255 (inclusive), o que pode não ser o caso de ***x***, ***y*** e ***z***. *DICA: sua função do item anterior já faz isso!*

RESPOSTA:
```
cor novaCor(int x, int y, int z) {
      cor c = {nivel(x), nivel(y), nivel(z)};
      return c;
}
```

d) **(1.5 ponto)** Crie uma **função** que receba duas **cores** de entrada e retorne a cor correspondente à soma delas, onde cada canal é a soma dos respectivos canais das cores de entrada (i.e. ***r*** = soma dos vermelhos, ***g*** = soma dos verdes, ***b*** = soma dos azuis). *DICA: lembre-se que o valor máximo por canal é 255 e que você já possui uma função para tratar isso!* **Por exemplo:** Para 2 cores *c1* e *c2* de entrada, onde
`r1 = 0;` `g1 = 130; b1 = 245` e
`r2 = 80;` `g2 = 100; b2 = 55,` a cor retornada terá os valores `r = 80; g = 230; b = 255`.

RESPOSTA:
```
cor soma(cor c1, cor c2) {
      cor c = { nivel(c1.r + c2.r), nivel(c1.g + c2.g), nivel(c1.b + c2.b) };
      return c;
}
```

e) **(2.0 pontos)** Crie uma **função** que receba um vetor de **cores** e seu tamanho N, e então (i) ordena esse vetor em ordem **crescente** do canal ***r*** (vermelho) de cada cor pelo método **Bubble Sort** e (ii) imprime todas as cores no formato "(r, g, b)", **seguindo a ordenação crescente estabelecida no item (i)**. Sua função **não** deve retornar valores.

RESPOSTA:

```
void ordenaEImprime(cor v[], int n) {
	const int V = 1, F = 0; // Verdadeiro / Falso
	int fim = n - 1;
	int ordenado = F;
	while (!ordenado) {
		ordenado = V;
		for (int i = 0; i < fim; i++){
			if (v[i].r > v[i+1].r) {
				cor aux = v[i];
				v[i] = v[i + 1];
				v[i + 1] = aux;
				ordenado = F;
			}
		}
		--fim;
	}
	//Impressão:
	for (int i = 0; i < n; i++)
		printf("(%d, %d, %d)", v[i].r, v[i].g, v[i].b);
}
```

Questão 5) (2.0 pontos) O que será impresso ao final de cada iteração do programa abaixo? No seu caderno de respostas, faça uma tabela análoga à descrita a seguir, e a preencha com os valores esperados:

```
#include <stdio.h>
int main() {
	float a = 10, b = 2;
	int x = 0;
	for (int i = 1; i <= 3; i++) {
		b += i + i / 2;
		a = i + b / 2;
		x = b - a;
		if (b && x) {
			printf("a = %.1f; b = %.1f; x = %d\n", a, b, x);
			continue;
			b = 0;
		}
		a = 1 / 2;
		printf("a = %.1f; b = %.1f; x = %d\n", a, b, x);
	}
	return 0;
}
```

i	a	b	x
1	?	?	?
2	?	?	?
3	?	?	?

RESPOSTA:

```
a = 0.0; b = 3.0;  x = 0
a = 5.0; b = 6.0;  x = 1
a = 8.0; b = 10.0; x = 2
```