FEEC - UNICAMP;
EA871;
EXPERIMENTO DA TAREFA 1;
VINÍCIUS ESPERANÇA MANTOVANI, RA: 247395

ÍTEM 1:

	Python (tipo de dado, visibilidade)	C (tipo de dado, visibilidade)	Entrada/Saída/ de Trabalho
Problema 12	n - int - local m - int - local anterior - int - local atual - int - local resto - int - local	anterior - int - local atual - int - local resto - int - local	Entrada Python: n, m; Entrada C: anterior, atual; Saída Python: n, m, atual; Saída C: anterior Trabalho Python: anterior, resto; Trabalho C: resto
Problema 3	n - int - local i - int - local x - float - local y - float - local	i - int - local n - int - local x - float - local y - float - local	Entrada Python: n; Entrada C: n; Saída Python: x, y; Saída C: x, y; Trabalho Python: i; Trabalho C: i;
Problema 6	frase - string - local palavra - string - local m - int - local n - int - local numOcor - int - local	ocorre - char - local m - int - local n - int - local i - int - local numOcor - int - local frase - char[] - local palavra - char[] - local	Entrada Python: frase, palavra; Entrada C: frase, palavra; Saída Python: frase, palavra, m e n; Saída C: frase, palavra, m, n, numOcor; Trabalho Python: m, n, numOcor; Trabalho C: m, n, ocorre, numOcor,

a) Não há variáveis globais em nenhum dos programas.

c) O tamanho alocado para a variável é maior em Python, uma vez que, por ser uma linguagem de alto nível, interpretada, suas variáveis são parte de um objeto, abstração da memória que armazenam tanto a variável como funções que podem ser aplicadas aos valor da variável.

d) A solução em Python equivalente àquela aplicada em C é a solução 3, já que ambas as soluções (em Python e C) trabalham com um loop dependente da variável resto para calcular o mdc, baseado no resto dado na divisão de anterior por atual, apesar de, em C o resto iniciar com valor igual ao resto da divisão de atual por anterior, enquanto que em Python o valor de resto já inicia com sendo anterior % atual.

ÍTEM 2:

	Nome do Operador	Python	C
a & b	AND	100	100
a | b	OR	100	100
a ^ b	XOR	0	0
~a	NOT	-101	155
~b	NOT	-101	-101
a << 2	LEFT SHIFT	400	144
a >> 2	RIGHT SHIFT	25	25

b) A diferença entre os valores impressos de ~a e ~b ocorre por causa que, o ~a, antes de ser impresso, é armazenado na unsigned int c, o que faz com que seja impresso como um número sem sinal, enquanto que o ~b é impresso diretamente, o que o faz ser impresso com sinal.

ÍTEM 3:

Sim, a ordem de geração do .o e do .elf é condizente com o que é mostrado na figura 1 do roteiro. Isso porque, o .o é gerado antes do .elf no CodeWarriors, assim como mostrado na figura 1..

ÍTEM 4:

a) RESULTADOS:

UINT8_T:

n = 5 → 120 → certo apenas até o n = 5
n = 6 → 208
n = 7 → 176
n = 8 → 128
n = 9 → 128

n = 10 → 0
n = 11 → 0
n = 12 → 0
n = 13 → 0

UINT16_T:
n = 5 → 120
n = 6 → 720
n = 7 → 5040
n = 8 → 40320 → certo até o n = 8
n = 9 → 35200
n = 10 → 24320
n = 11 → 5376
n = 12 → 64512
n = 13 → 52224

UINT32_T:
n = 5 → 120
n = 6 → 720
n = 7 → 5040
n = 8 → 40320
n = 9 → 362880
n = 10 → 3628800
n = 11 → 39916800
n = 12 → 479001600 → certo até o n = 12
n = 13 → 1932053504

INT8_T:
n = 5 → 120 → certo apenas até o n = 5
n = 6 → -48
n = 7 → -80
n = 8 → -128
n = 9 → -128
n = 10 → 0
n = 11 → 0
n = 12 → 0
n = 13 → 0

INT16_T:
n = 5 → 120
n = 6 → 720
n = 7 → 5040 → certo até n = 7
n = 8 → -25216
n = 9 → -30336
n = 10 → 24320
n = 11 → 5376
n = 12 → -1024
n = 13 → -13312

INT32_T:

n = 5 → 120
n = 6 → 720
n = 7 → 5040
n = 8 → 40320
n = 9 → 362880
n = 10 → 3628800
n = 11 → 39916800
n = 12 → 479001600 → certo até o n = 12
n = 13 → 1932053504

	uint8_t	uint16_t	uint32_t	int8_t	int16_t	int32_t
n = 5	120	120	120	120	120	120
n = 6	208	720	720	-48	720	720
n = 7	176	5040	5040	-80	5040	5040
n = 8	128	40320	40320	-128	-25216	40320
n = 9	128	35200	362880	-128	-30336	362880
n = 10	0	24320	3628800	0	24320	3628800
n = 11	0	5376	39916800	0	5376	39916800
n = 12	0	64512	479001600	0	-1024	479001600
n = 13	0	52224	1932053504	0	-13312	1932053504

4.b) Os tipos sem sinal conseguem armazenar números maiores que os tipos com sinal, pois, por possuírem um de seus bits destinado a mostrar o sinal do número, os tipos com sinal têm um bit a menos para armazenar o valor numérico desejado. Logo, os tipos com sinal acabam comportando valores menores que os sem sinal.

4.c) Os tipos de dados impactam nos resultados, por causa de afetarem o número de bits usados para comportar os valores. Isso porque, apesar de na memória os bits restantes serem completados com 0, quanto menor o tipo especificado de dado em quantidade de bits, menor o valor máximo que pode ser armazenado nesse espaço de memória. Desse modo, tem-se que, se o valor a ser armazenado for maior que o número de bits do tipo especificado, ocorre que parte dos bits são desprezados, alterando o valor que seria o correto.