QUESTÃO 01

MAT0: 11511ECP011 MAT1: 11621ECP016 MAT2: 11711ECP012  
KANO0 = 1 KANO1 = 3 KANO2 = 3  
KCUR0 = 3 KCUR1 = 3 KCUR2 = 3  
KNUM0 = 3 KNUM1 = 8 KNUM2 = 4

QUESTÃO 02

As constantes apresentadas são:

X = 11  
Y = 3  
Z = 1

Logo tivemos um resultado do tipo:

11 3 1

QUESTÃO 03

As constantes apresentadas são:

X = 0  
Y = 4  
Z = 9

Logo tivemos um resultado do tipo:

0 4 9

QUESTÃO 04

Nesta questão temos o uso de um somatório para fazer a operação de multiplicação, mas antes de se fazer o somatório a mesma variável é usando em uma soma de outras duas variáveis, logo para termos um resultado preciso da multiplicação precisaríamos “zerar” a variável soma antes ou na própria inicialização do laço *“for”*, que foi o jeito que eu adotei para resolver o bug.

int x = KANO0, y = KANO1+KANO2, z = KNUM0+KNUM1+KNUM2;  
int i, soma = x+y;  
printf("A soma de %d e %d = %d\n", x, y, soma);  
if(z > 10)  
printf("O numero z = %d (maior que 10)\n", z);  
for(i = 0, **soma = 0**; i < x; i++)  
soma = soma + z;  
printf("O resultado de %d vezes %d = %d\n", x, z, soma);

QUESTÃO 05

1. Os números informados no código foram:

69 0 80

1. A natureza desses números são os seus códigos de caracteres decimais contidos na tabela ASCII, por exemplo, temos o 69 = E, 0 = NULL e 80 = P. Esses códigos foram obtidos quando se tem uma posição em um vetor de “*string”* e coloca como especificador o “%d”, que é o especificador dos números inteiros. Com isso o compilador mostra o seu código ASCII do carácter gravado naquela posição.

QUESTÃO 06

1. A tigresa foi preso por um dragao.
2. Temos “*strings”* armazenando um conjunto de caracteres, onde que através das constantes geradas no exercício 1 temos variáveis de controles “x, y e z” as quais são valores da posição dos vetores do tamanho da mensagem mostrada e além disso a posição de inicio e fim de parte da mensagem. Temos um ponteiro apontando também para cada *string* em cada *for* onde ele é do tipo *char*, ou seja, ele faz a varredura das *strings byte* a *byte* onde conseguimos ter uma maior manipulação dos dados, mas ao invés de mostrar o correspondente da *string* ele “diminui” um caractere, em linhas gerais podemos dizer que as mensagens estão gravadas nas *strings* sempre com um caractere que sucede, logo quando vai fazer a impressão o ponteiro pega cada um dos caracteres referentes a posição dos vetores de controle de tamanho e imprime seu antecessor, e isso tem se como base a tabela ASCII