Universidade Federal Fluminense
Campus de Rio das Ostras
Instituto de Ciência e Tecnologia

Programação de Computadores II
Verificação de Reposição (VR) 02/12/2019
Prof. Alessandro Copetti

Nome: _____**GABARITO**_____

Obs.: A prova pode ser feita a lápis! As funções não devem conter valores fixos!

1. O que será impresso na tela? Mostre o valor que cada variável assume durante a execução e também circule a resposta; não é para explicar o que cada linha faz.

a) (1pt)

```
main() {
   int matriz[3][3]={{1,2,3},{4,5,6},{7,8,9}};
   int i,j,soma=0;
   for (i=0;i<3;i++)
      for(j=0;j<3;j++)
          if (i==j)
              soma=soma+matriz[i][j];
   printf("\n%d",soma);
}
```

15

b) (1,5pt)

```
main(void) {
   char str[50]="uma frase de teste";
   char str2[50];
   int i,j;
   for (i=0,j=0;i<7;i=i+2,j++)
       str2[j]='a'+i;
   str2[j]='\0';

   printf("\n%s", str2);

   strcpy(str2,str);

   str[3]='\0';
   printf("\n%d", strlen(str));
```

aceg
3

c) (2pts)

```
typedef struct {
    int x, y;
} Ponto;
main() {
    int i=0, var=0;
    Ponto ponto[3]={{1,2},{3,4},{5,6}};
    Ponto **p = (Ponto **) malloc(sizeof(Ponto *));
    p[1]=&ponto[1];
    p[1]->x++;
    p[1]->y++;
    for (i=0;i<3;i++)
        var = var + ponto[i].x + ponto[i].y;
    printf("%d", var);


    Ponto pts[3]={{1,2},{3,4},{5,6}};
    printf("\n%d",funcao(pts,3));
}
```

```
int funcao(Ponto *pt, int n) {
    int i=0;
    for (;i<n;i++)
        pt[i].x = pt[i].x + 2;
    return pt[n-1].x % pt[n-2].x;
}
```

23
2

d) (2pts)

```
typedef struct tlista{
    int id;
    struct tlista *prox;
}TLista;

TLista* adiciona(TLista *lista, int valor){
    TLista* novo = (TLista*)malloc(sizeof(TLista));
    novo->id = valor;
    novo->prox = lista;
    return novo;
}

TLista* funcao(TLista* l1, TLista* l2){
    TLista* lista = NULL;
    TLista* aux = l1;
    TLista* aux2 = l2;

    while(aux!= NULL){
        while(aux2 != NULL){
            if(aux->id == aux2->id)
                lista = adiciona(lista, aux->id);
            aux2 = aux2->prox;
        }
        aux = aux->prox;
        aux2 = l2;
    }
    return lista;
}

void imprime(char *titulo, TLista *l){
    TLista *aux=l;
    printf("\n%s:\n", titulo);
    while (aux!=NULL){
        printf("%d\n", aux->id);
        aux = aux->prox;
    }
}

int main(){
    TLista* tlista1 = NULL;
    TLista* tlista2 = NULL;
    int i;
    for(i=0; i<4; i++){
        tlista1 = adiciona(tlista1, i);
        tlista2 = adiciona(tlista2, i*2);
    }
    TLista* tlista3 = NULL;
    tlista3 = funcao(tlista1, tlista2);

    imprime("Lista 1",tlista1);
    imprime("Lista 2",tlista2);
    imprime("Lista 3",tlista3);
}
```

Lista 1:
3
2

1
0

Lista 2:
6
4
2
0

Lista 3:
0
2

2. (2pts) Desenvolva uma função que cria, inicializa com zero e retorna uma matriz. A função recebe o número de linhas e de colunas da matriz desejada. O protótipo é:

```
int **alocaInicializaMatriz(int lin, int col);
```

```
int **alocaInicializaMatriz(int lin, int col) {
   int i,j;
   int **matriz = (int **) malloc(lin*sizeof(int *));
   for (i=0;i<lin;i++) {
       matriz[i] = (int *) malloc(col*sizeof(int));
       for (j=0;j<col;j++)
          matriz[i][j]=0;
   }
   return matriz;
}
```

3. Tendo o seguinte código:

```
#define N 10

main() {
   int v1[N]={1,2,3,4,5,6,7,8,9,10}, *v2, numElemImpares,i;

   v2 = impares(v1, N, &numElemImpares);
   for (i=0; i<numElemImpares; i++)
       printf("%d ",v2[i]);
}
```

a. (2,5pts) Desenvolva a função `impares()` que retorna um vetor contendo somente os elementos ímpares do vetor passado como parâmetro. A função deve ser chamada de acordo com o que está apresentado na main.

```
int *impares(int *vet, int n, int *numImpares){
   int i=0, cont=0, *vetorImpares;
   for(;i<n;i++)
      if (vet[i]%2!=0) cont++;
   *numImpares = cont;

   vetorImpares = (int *) malloc (cont * sizeof(int));
   for(cont=0,i=0;i<n;i++)
      if (vet[i]%2!=0) vetorImpares[cont++]=vet[i];
   return vetorImpares;
}
```