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

Programação de Computadores II
Verificação de Reposição (VR) 07/12/2017
Prof. Alessandro Copetti

Nome: _____**GABARITO**_____

Obs.: A prova pode ser feita a lápis!

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)

```
#include <stdio.h>
main() {
    int i, space, rows=3, k=0;

    for(i=1; i<=rows; i++, k=0) {
        for(space=1; space<=rows-i; space++)
            printf("  ");

        while(k != 2*i-1) {
            printf("* ");
            k++;
        }

        printf("\n");
    }
}
```

```
    *
  * * *
* * * * *
```

b) (1pt)

```
int main () {
  char sentence []="Maria tem 12 anos de idade";
  char str [20], palavra[20];
  int i;

  sscanf (sentence,"%s %s %d",str,palavra,&i);
  printf("\n%d", i++);
}
```

12

c) (1pt)

```
void f1(int *p) {
   *p = *p * 2;
}

void f2(int x, int *p) {
    x = x + 2;
   *p = *p * x;
}

main() {
   int x=2,y=6,*p=&y;
   f1(&x);
   f2(x,p);
   printf("%d",x+y);
}
```

40

2. Tendo o código abaixo e em seguida a execução do programa:

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

```
#include <stdlib.h>
#include <string.h>

char **alocaTexto(char *nomeArq,int *linhas) {
    char **texto, c;
    char linha[121];
    int nlinhas=0,i,tam=0;

    FILE *fp=fopen(nomeArq,"r");
    while ((c = fgetc(fp)) != EOF) {
        if (c == '\n')
            nlinhas++;
    }
    *linhas = nlinhas;
    rewind(fp); // vai para inicio do arquivo

    texto = (char **) malloc(nlinhas*sizeof(char *));
    for (i=0;i<nlinhas;i++) {
         fgets(linha,120,fp);
         tam=strlen(linha);
         texto[i]=(char *) malloc((tam+1)*sizeof(char));
         strncpy(texto[i],linha,tam-1);
         texto[i][tam]='\0';
    }
    fclose(fp);
    return texto;
}

void imprime(char **texto, int linhas) {
   int i;
   for (i=0;i<linhas;i++)
       printf("\n%s",texto[i]);
}

main() {
    char **texto;
    int *vetVogais, numLinhas=0, i;
    texto = alocaTexto("texto.txt", &numLinhas);
    imprime(texto, numLinhas);

    vetVogais = linhasVogais(texto, numLinhas);
    for (i=0; i<numLinhas;i++)
        printf("\nlinha %d = %d vogais", i, vetVogais[i]);

}
```

```
para um texto qualquer
a funcao a ser desenvolvida
deve retornar
o numero de vogais
em cada linha do texto
linha 0 = 9 vogais
linha 1 = 11 vogais
linha 2 = 5 vogais
linha 3 = 8 vogais
linha 4 = 8 vogais
```

a. (3pts) Desenvolva a função linhasVogais que deve retornar um novo vetor com o número de vogais encontradas em cada linha do arquivo. Comentário: após fazer a condição para 'a' e 'e', vc pode somente colocar '...' para finalizar a linha do 'if'.

b. (3pts) Tendo a seguinte definição para uma lista simplesmente encadeada:

```
typedef struct listapalavras {
    char palavra[80];
    struct listapalavras *prox;
} ListaPalavras;
```

E a seguintes chamadas no fim da main, apresentada na questão '2a' acima:

```
    ListaPalavras *inicioLista = NULL;
    inicioLista = insereTodasPalavras(inicioLista, texto, numLinhas);
    imprimeLista(inicioLista);
```

Desenvolva a função insereTodasPalavras, a qual pega cada palavra que está na matriz *texto* e insere no início da lista ligada.
A função imprimeLista(inicioLista) deve imprimir na tela:

```
Palavras na Lista:
texto do linha cada em vogais de numero o retornar deve desenvolvida
 funcao a qualquer texto um para
```

A função imprimeLista possui o seguinte código:

```
void imprimeLista(ListaPalavras *inicioLista) {
    int i;
    printf("\nPalavras na Lista:");
    ListaPalavras *p = inicioLista;
    for (i=0; p!=NULL; p=p->prox)
        printf("\n%s", p->palavra);
}
```

Dica para extrair uma palavra de 'texto':

```
     strncpy(palavra, &texto[i][inicioPalavra],j-inicioPalavra);
     palavra[j-inicioPalavra]='\0';
     inicioPalavra = j+1;
     inicioLista = insereInicio(inicioLista, palavra);
```

c. (1pt) Desenvolva a função insereInicio, a qual insere a palavra fornecida no início da lista ligada. O protótipo dessa função deve ser:

```
ListaPalavras *insereInicio(ListaPalavras *inicio, char palavra[80]);
```

```
int *linhasVogais(char **texto, int linhas) {
   int i,j,tam;
   int *vetVogais=(int *) malloc(linhas * sizeof(int));
   for (i=0;i<linhas;i++) {
      vetVogais[i]=0;
      tam=strlen(texto[i]);
      for (j=0; j<tam; j++)
         if ((texto[i][j]=='a') || (texto[i][j]=='e') || (texto[i][j]=='i') || (texto[i][j]=='o') ||
(texto[i][j]=='u'))
            vetVogais[i]++;
   }
   return vetVogais;
}

ListaPalavras *insereInicio(ListaPalavras *inicio, char palavra[80]) {
   ListaPalavras *novo = (ListaPalavras *) malloc(sizeof(ListaPalavras));
   strcpy(novo->palavra, palavra);
   novo->prox = inicio;
   return novo;
}

ListaPalavras * insereTodasPalavras(ListaPalavras *inicioLista, char **texto, int linhas) {
   int i, j, inicioPalavra;
   char palavra[80];

   for (i=0; i<linhas; i++) {
      // pega uma palavra
      inicioPalavra = 0;
      for (j=0; texto[i][j]!='\0'; j++)
         if (texto[i][j]==' ') {
            strncpy(palavra, &texto[i][inicioPalavra],j-inicioPalavra);
            palavra[j-inicioPalavra]='\0';
            inicioPalavra = j+1;
```

```
            inicioLista = insereInicio(inicioLista, palavra);
        }
      strncpy(palavra, &texto[i][inicioPalavra],j-inicioPalavra);
      palavra[j-inicioPalavra]='\0';
      inicioLista = insereInicio(inicioLista, palavra);
   }
   return inicioLista;
}
```