Programação Imperativa / Programação de Computadores – PARTE 1 (30m)

Exame fictício

Nome:	
NMEC:	**BInº:**

Notas importantes:

- **Tempo disponível: 30m para a 1ª parte, 60m para a 2ª parte.**
- ***Notas.* Há classificação mínima da 1ª parte. Classificação mínima no exame: 7.5 em 20.**

---

1. Como é implementado um "string" em linguagem C?

   [ ] Vector de caracteres terminado com `EOF`.
   [X] Vector de caracteres terminado com `0`.
   [ ] Vector de caracteres terminado com `'0'`.
   [ ] Vector de inteiros iniciado com `-1`.
   [ ] Vector de inteiros terminado com o código ASCII de `'0'`.
   [ ] Vector de macros terminado com `'\000'`.

---

2. O que imprime o seguinte programa?

```
int func(int v[],int n){
  int s=0, i;
  for(i=0;i<n-1;i++)
    v[i+1]=2*v[i];
  return v[n-1];
}
main(){
  int m=5, x[]={1,2,9,6,0};
  printf("%d ",func(x,m));
}
```

   [ ] 8 [ ] 0 [X] 16 [ ] 1 [ ] 12

Programação Imperativa / Programação de Computadores – PARTE 2 (60m)
Exame fictício

Nome:	
NMEC:	**BInº:**

Notas importantes:

- **Tempo disponível: 30m para a 1ª parte, 60m para a 2ª parte.**
- **Todos os programas e funções devem ser escritos em linguagem C.**
- **Os programas e funções devem ser claros e concisos. Quando achar conveniente inclua comentários que facilitem a sua compreensão: notas sobre o método utilizado e significado das principais variáveis.**
- **Não pode utilizar funções de "strings" pré-definidas (`strcmp`, `strcpy`, etc.)**

---

1. Escreva uma função

   `void duplica(char s[])`

   cujo efeito é duplicar os caracteres do "string" `s`.
   Exemplo. O "string" `"x + 12"` é transformado em `"xx  ++  1122"`
   Nota. Supõe-se que existe espaço suficiente em `s`.

   Resposta:

```
\begin{verbatim}
void duplica(char s[]){
  //-- o caracter s[i] vai para s[2i] e para s[2i+1]
  int i=0;
  while(s[i]) i++;
  s[2*i]=0;
  i--;
  while(i>=0){
    s[2*i]  =s[i];
    s[2*i+1]=s[i];
    i--;
  }
}
```

2. *Ajustando uma linha*
Um "string" `s` contém apenas letras, espaços e sinais de pontuação como "," e ".". Uma *palavra* é uma sequência máxima de caracteres diferentes de espaço. Por exemplo o "string" `" Ai, as 3 moscas!"` contém 4 palavras, `"Ai,"`, `"as"`, `"3"` e `"moscas!"` e 4 espaços.

Escreva uma função

```
void ajusta(char *s, int larg)
```

cujo efeito é *justificar* a linha de texto representada pelo "string", isto é, espaçar as palavras por forma que a primeira esteja encostada à esquerda, a última à direita e de modo que o número de espaços entre 2 palavras consecutivas seja aproximadamente igual (só pode diferir no máximo de uma unidade). O inteiro `larg` representa a largura da linha em número de caracteres.
Se a justificação não for possível, o "string" deve ficar inalterado.

Exemplo. Para o "string" exemplificado e para `larg=20`, poderia ficar

```
i:      0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
        ------------------------------------------------------------
s[i]    A  i  ,           a  s           3        m  o  s  c  a  s  !
```

Os espaçamentos são sucessivamente 3, 3 e 2 (também poderiam ser, por exemplo, 3, 2, 3 mas não 4, 2, 2).

Notas importantes. (i) Admite-se que `s` contém espaço suficiente, isto é, pelo menos menos `larg+1` caracteres. (ii) Admite-se que `s` contém pelo menos 2 palavras (o que implica conter pelo menos um espaço). (iii) Escreva a sua função e forma clara, escrevendo se achar conveniente funções auxiliares; dê uma breve explicação do método utilizado.

(Ver página seguinte)

Resposta
Embora não fosse pedido, incluimos o `main` para que o aluno possa testar o programa.

```
#include <stdio.h>
#define MAX  200
#define MAXP 20

/*-------------------------------------------------------
  "medicoes"
   - calcula: o inicio de cada palavra, o seu comprimento,
       o número de palavras
   - retorna o número de palavras                        */
int medicoes(char s[],int inicia[],int compr[]){
  int i=0, pal=0, conta=0,dentro=0;
  for(i=0;s[i];i++)
    if(dentro && s[i]==' '){         //-- acabou palavra
      compr[pal]=conta; dentro=0; pal++;
    }
    else if(dentro && s[i+1]==0){    //-- palavra até ao fim de s
      compr[pal]=conta+1; pal++;
      }
      else if(!dentro && s[i]!=' '){ //-- começa palavra --
          inicia[pal]=i;compr[i]=1;conta=1;dentro=1;
        }
        else if(s[i]!=' ') conta++;  //-- +1 => comprimento
  return pal;
}
/*-------------------------------------------------------
  Desloca cada palavra para o local correcto
  Usa o string "temporário t[]
*/
void movimenta(char s[],int larg, int inicia[],int compr[],
          int npals, int sep, int num1){
  int  i,ind,p;
  char t[MAX];
  for(i=0;s[i];i++) t[i]=s[i];  //--    s => t
  for(i=0;i<larg;i++) s[i]=' '; //--  ' ' => s
  s[larg]=0;
  ind=0;
  for(p=0;p<npals;p++){         //-- copia cada palavra:
    for(i=0;i<compr[p];i++)
      s[ind++]=t[inicia[p]+i];  //--   para s
    ind += sep + (p<num1?1:0);
  }
}
//-------------------------------------------------------
void ajusta(char s[], int larg){
  int inicia[MAXP],     //-- inicio de cada palavra
      compr[MAXP],      //-- comprimento de cada palavra
      i,soma,livre,sep,num1,npals;
  npals = medicoes(s,inicia,compr);
  if(npals<=1) return;  //-- Impossivel justificar
  soma=0;               //-- Calcula soma dos comprimentos
  for(i=0;i<npals;i++) soma += compr[i];
  livre=larg-soma;      //-- total de espaços
  sep =livre/(npals-1); //-- separação
  num1=livre%(npals-1); //-- no. palavras com espaçamento sep+1
  movimenta(s,larg,inicia,compr,npals,sep,num1);
}
//-------------------------------------------------------
int main(){
  char s[25]=" Ai, as 3 moscas!";
  printf("  |%s|\n",s);
  ajusta(s,20);
  printf("  |%s|\n",s);
  return 0;
}
```