Programação I
Segunda prova - Correção

1. (**4 pontos)** Suponha que seu programa tenha uma função que leia um texto longo do teclado, armazenando-o numa variável do tipo cadeia de caracteres. Esse texto pode conter vários espaços em branco entre cada palavra, além de vários caracteres '\n'. Escreva uma função que copie esse texto em outra variável cadeia de caracteres, deixando apenas um branco entre cada palavra e deixando os '\n' com no mínimo 55 caracteres e no máximo 65 caracteres entre dois deles, mas sempre o mais próximo possível de 65 caracteres.

```
void formata_texto(char in[10000], char out[10000])
{int i=0, j=0, cont=0;
 char aux[20], ending[2];

  ending[0]='\n';   ending[1]='\0';  // serve para terminar linhas
  while (in[i] != '\0') // laço até terminar o texto original
  { while ((in[i] != ' ') && (in[i] != '\0'))  // laço para copiar
    { if (in[i] != '\n')          // palavra por palavra
          { aux[j] = in[i];
            j++;   // se copiar o caracter, então avance o valor de j
          }
      i++;   // avança o valor de i qualquer que seja o caracter lido
    }
    if (in[i] != '\0')   // se já é o final do texto pula tudo
        { aux[j]='\0';  // terminar a string com a palavra
          if (cont + strlen(aux) <= 65)  // se ainda cabe na linha
              { strcat(out, aux);   // copia palavra e ajusta cont
                strcat(out, " ");
                cont = cont + strlen(aux) + 1;
              }
            else    // se não cabe mais na linha então termina linha
              { strcat(out, ending);  // atual e copia palavra na
                strcat(out, aux);   // proxima linha, reajustando o
                strcat(out, " ");   // valor de cont
                cont = strlen(aux) + 1;
              }
          while ((in[i] == ' ') || (in[i] == '\n'))  // limpa os demais
             i++;     // brancos e \n até a próxima palavra
          j = 0;
        }
  }
  strcat(out," ");  // termina a frase com um último branco, para
                    // colocar o \0 no final da string de resultado
}  /* Observar que variações nesse código são perfeitamente possíveis.
Essa é uma possível implementação.....  */
```

2. (**3 pontos**) Dois problemas associados ao processo de ordenação de elementos num vetor são o de inserção de novos elementos no vetor e o de busca de um elemento nesse vetor. Escreva duas funções que recebam um vetor de números reais, seu tamanho e um número real, e realizem as operações indicadas acima. No caso da busca, sua função deve retornar o índice da posição em que o número foi encontrado ou -1 caso contrário. No caso da inserção, deve inserir o novo número na posição correta.

```
/* busca pelo número até que o valor armazenado no vetor
   seja maior ou igual ao número procurado ou ainda que se
   passe por todos os números do vetor (i==tam)  */
int busca(int tam, float vetor[10000], float quem)
{int i=0;

  while ((vetor[i] < quem) && (i < tam))  i++;
  if (vetor[i] == quem)
      return(i);
    else
      return(-1);
}

/* insere o novo valor deslocando os valores maiores
   que ele uma posição adiante no vetor  */
void insere(int tam, float vetor[10000], float quem)
{int i=0, j;

  while ((vetor[i] <= quem) && (i < tam)) i++;
  for (j=tam; j>i; j--)
     vetor[j] = vetor[j-1];
  vetor[j] = quem;
}
```

3. (**3 pontos**) Considere o seguinte trecho de programa:

```
main()
{ int a=1, b=2, c=3;
  float x=0.5, y;
        y = func1 (&x, &a);
        printf (" a=%d   x=%f   y=%f\n", a, x, y);
=======>>>>>>   IMPRIME   "a=1  x=3.5000   y=1.2500"   <<<<<<<======
        b = func2 (a, &c);
        printf (" a=%d   b=%d   c=%d\n", a, b, c);
=======>>>>>>   IMPRIME   "a=1  b=2   c=1"             <<<<<<<======
        x = func3 (c, a);
        printf (" a=%d   c=%d   x=%f\n", a, c, x);
=======>>>>>>   IMPRIME   "a=1  c=1   x=0.0000"        <<<<<<<======
}
```