| INF 1620 – P1 - 10/04/02 | Questão 1 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere uma disciplina que adota o seguinte critério de aprovação: os alunos fazem duas provas (P1 e P2) iniciais; se a média nessas duas provas for maior ou igual a 5.0, e se nenhuma das duas notas for inferior a 3.0, o aluno passa direto. Caso contrário, o aluno faz uma terceira prova (P3) e a média é calculada considerando-se essa terceira nota e a maior das notas entre P1 e P2. Neste caso, o aluno é aprovado se a média final for maior ou igual a 5.0.

Escreva um programa completo que leia inicialmente as duas notas de um aluno, fornecidas pelo usuário via o teclado. Se as notas não forem suficiente para o aluno passar direto, o programa deve capturar a nota da terceira prova, também fornecida via o teclado. Como saída, o programa deve imprimir a média final do aluno, seguida da mensagem "Aprovado" ou "Reprovado", conforme o critério descrito acima.

```
#include <stdio.h>

int main (void)
{
   float p1, p2, p3;   /* notas das provas */
   float m;            /* media final */

   /* captura as duas primeiras notas */
   printf("Digite as notas da P1 e da P2: ");  /* opcional */
   scanf("%f %f", &p1, &p2);

   /* calcula media */
   m = (p1 + p2) / 2;

   /* verifica se precisa da P3 */
   if (m<5.0 || p1<3.0 || p2<3.0) {

      /* captura a terceira nota */
      printf("Digite a nota da P3: ");  /* opcional */
      scanf("%f", &p3);

      /* calcula nova media */
      if (p1 > p2)
         m = (p1 + p3) / 2;
      else
         m = (p2 + p3) / 2;
   }

   /* imprime saida */
   if (m < 5.0)
      printf("%f   Reprovado\n", m);
   else
      printf("%f   Aprovado\n", m);

   return 0;
}
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P1 - 10/04/02 | Questão 2 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere uma função, chamada `somaprod`, que calcula o somatório e o produtório dos elementos de um vetor qualquer de valores reais. O programa abaixo ilustra a utilização dessa função. Se fosse executado, este programa imprimiria os valores `16.70` e `237.60`, que representam, respectivamente, a soma e o produto dos elementos do vetor em questão.

```
#include <stdio.h>
int main (void) {
   float v[5] = {3.2, 4.5, 2.0, 5.5, 1.5};
   float s, p;
   somaprod( 5, v, &s, &p ); /* chama somaprod */
   printf("%.2f  %.2f\n",s,p);
   return 0;
}
```

Escreva a função `somaprod` para que o programa acima funcione de maneira correta.

*A função deve receber como parâmetros o número de elementos do vetor, o endereço do primeiro elemento do vetor, o endereço da variável que representará o somatório e o endereço da variável que representará o produtório.*

```
void somaprod (int n, float* v, float *ps, float *pp)
{
   int i;

   /* inicializa variaveis */
   *ps = 0.0;
   *pp = 1.0;

   /* calcula somatorio e produtorio */
   for (i=0; i<n; i++) {
      *ps = *ps + v[i];
      *pp = *pp * v[i];
   }

}
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P1 - 10/04/02 | Questão 3 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Escreva uma função em C para converter uma cadeia de caracteres para letras minúsculas. A função deve receber como parâmetro de entrada uma cadeia de caracteres e ter como valor de retorno uma nova cadeia, cujo espaço de memória deve ser alocado dinamicamente pela função, contendo a cadeia original convertida para letras minúsculas. Por exemplo, se for passada a cadeia de caracteres "Puc-Rio", a função deve retornar a cadeia "puc-rio". A assinatura da função deve ser:

```
char* converte (char* s);
```

*Dica*: Se `c` representa um caractere maiúsculo, o caractere minúsculo correspondente pode ser dado por: `c-'A'+'a'`.

```
char* converte (char* s)
{
   int i;
   char* m;

   /* aloca memória para a nova cadeia */
   m = (char*) malloc (strlen(s) + 1);

   /* atribui elemento a elemento,
      convertendo para minúscula quando necessário */
   for (i=0; s[i]!='\0'; i++) {

      if (s[i]>='A' && s[i]<='Z')    /* verifica se é maiúscula */
         m[i] = s[i] - 'A' + 'a';    /* converte para minúscula */
      else
         m[i] = s[i];                /* atribui sem conversão */

   }
   m[i] = '\0';    /* marca fim da cadeia */

   return m;

}
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P1 - 10/04/02 | Questão 4 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Um ponto geométrico no espaço 2D é representado pelas coordenadas reais *x* e *y*. Considere uma aplicação que precisa manipular pontos, através da implementação de duas funções. A primeira recebe valores *x* e *y* e cria dinamicamente um tipo ponto correspondente. A segunda recebe dois pontos como parâmetros e retorna a distância entre eles. As assinaturas dessas funções são apresentadas abaixo:

```
/* definição de tipo */
typedef struct ponto *Ponto;

/* funções */
Ponto cria (double x, double y);
double distancia (Ponto a, Ponto b);
```

Escreva um código C que defina a estrutura `struct ponto` e faça a implementação das duas funções.

```
/* definicao da estrutura */
struct ponto {
   double x;
   double y;
};

/* funcao cria */
Ponto cria (double x, double y)
{
   Ponto p;

   /* aloca ponto e atribui campos */
   p = (Ponto) malloc(sizeof(struct ponto));
   p->x = x;
   p->y = y;

   return p;
}

/* funcao distancia */
double distancia (Ponto a, Ponto b)
{
   double d;

   d = sqrt((a->x - b->x)* (a->x - b->x) +
            (a->y - b->y)* (a->y - b->y));

   return d;
}
```

---

Formulário:

Fórmula da distância *d* entre dois pontos *a* e *b*: $d = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$

Função para extrair a raiz quadrada de um número (definida em math.h):

```
double sqrt (double);
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

| INF 1620 – P1 - 10/04/02 | Questão 5 |
|---|---|
| Nome: | |
| Matrícula: | Turma |

Considere uma matriz quadrada de dimensão `n x n`, onde `n` representa a dimensão da matriz. Essa matriz pode ser representada por um vetor de dimensão `n*n`, onde os elementos da primeira linha da matriz ocupam as primeiras posições do vetor, seguidos dos elementos da segunda linha, e assim por diante, conforme ilustrado esquematicamente abaixo:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \longrightarrow \begin{bmatrix} a & b & c & d & e & f & g & h & i \end{bmatrix}$$

Supondo que matrizes `n x n` de números inteiros são armazenadas em vetores, escreva uma função que, dada uma matriz, verifica se esta é ou não uma matriz triangular superior. A função deve retornar um se a matriz for triangular superior e zero se não for. O protótipo da função deve ser:

```
int superior (int n, int* mat);
```

*Nota*: Numa matriz triangular superior, todos os elementos abaixo da diagonal são iguais a zero.

*Vamos criar uma função auxiliar para acessar um elemento da matriz e então implementar a função que checa se é triangular superior.*

```
int acessa (int n, int* mat, int i, int j)
{
   int k = i*n + j;    /* indice para acessar vetor */
   return mat[k];
}

int superior (int n, int* mat)
{
   int i,j;

   /* testa os elementos abaixo da diagonal */
   for (i=1; i<n; i++) {
      for (j=0; j<i; j++) {
         if (acessa(n,mat,i,j) != 0)
            return 0; /* diferente de zero: não é triangular */
      }
   }
   return 1;      /* se chegou ao final dos loops, é triangular */
}
```

*Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.*

RASCUNHO

*Respostas nesta folha não serão consideradas.*

*Não separe as folhas deste caderno.*