INF 1620 – P3 - 02/07/02	Questão 1
Nome:	
Matrícula:	Turma

Considere um arquivo texto que contém os nomes e as notas dos alunos de uma disciplina. As informações de cada aluno ocupam duas linhas do arquivo: a primeira linha contém o nome do aluno (com eventuais espaços em branco) e a segunda as três notas obtidas. Considere que não existe linhas em branco no arquivo. Um exemplo deste formato é mostrado abaixo.

```
Fulano de Tal
6.3  5.7  7.1
Sicrano Silva
3.4  5.4  4.7
Beltrano Alves
9.2  6.8  8.3
```

Escreva um programa completo que leia o arquivo "entrada.txt", que contém as informações dos alunos no formato descrito acima, e imprima na tela o número de alunos aprovados e o número de alunos reprovados. Considere que um aluno é aprovado se obtiver média maior ou igual a cinco. Se não for possível abrir o arquivo, o programa deve ter como saída a mensagem "ERRO". Se não existir nenhum registro de aluno no arquivo (arquivo existente, mas vazio), deve-se imprimir valores zero. Para o exemplo ilustrado acima, a saída do programa seria:

```
Aprovados: 2
Reprovados: 1
```

```
#include <stdio.h>

int main (void)
{
   int a=0, r=0;
   char s[81];
   float p1, p2, p3;
   FILE* fp = fopen("entrada.txt","rt");

   if (fp==NULL) {
      printf("ERRO\n");
      return 1;
   }

   while (fscanf(fp," %[^\n] %f %f %f",s,&p1,&p2,&p3)==4) {
      float m = (p1+p2+p3)/3;
      if (m >= 5)
         a++;
      else
         r++;
   }
   fclose(fp);

   printf("Aprovados: %d\nReprovados: %d\n",a,r);
   return 0;
}
```

Protótipos de funções que podem ser úteis:

```
FILE* fopen (char* nome, char* modo);
int fclose (FILE* fp);
int fscanf (FILE* fp, char* formato, ...);
char* fgets (char* s, int n, FILE* fp);
int fgetc (FILE* fp);
```

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 – P3 - 02/07/02	Questão 2
Nome:	
Matrícula:	Turma

Considere o tipo que representa os dados associados aos alunos de uma disciplina dado por:

```
struct aluno {
   char nome[81];     /* nome do aluno */
   char turma;        /* turma do aluno */
   float p1, p2, p3;  /* notas obtidas */
};
typedef struct aluno Aluno;
```

Utilizando a função `qsort` da biblioteca padrão, escreva uma função para ordenar um vetor de estruturas `Aluno`. O vetor deve ser ordenado em ordem crescente de turma, usando os nomes, em ordem alfabética, como critério de desempate. A função recebe como parâmetros o número de alunos e o vetor a ser ordenado. O protótipo da função deve ser:

```
void ordena (int n, Aluno* vet);
```

Obs: a função `ordena` consistirá apenas da chamada da função `qsort`, fazendo uso de uma função auxiliar para comparar os elementos do vetor, que deve ser escrita.

```
int comp (const void* v1, const void* v2)
{
   Aluno* a1 = (Aluno*)v1;
   Aluno* a2 = (Aluno*)v2;
   if (a1->turma < a2->turma)
      return -1;
   else if (a1->turma > a2->turma)
      return 1;
   else
      return strcmp(a1->nome,a2->nome);
}

void ordena (int n, Aluno* vet)
{
   qsort(vet,n,sizeof(Aluno),comp);
}
```

Protótipos de funções que podem ser úteis:

```
int strcmp (char* s1, char* s2);
void qsort (void* vet, int n, int tam, int (*comp) (const void*, const void*));
```

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 – P3 - 02/07/02	Questão 3
Nome:	
Matrícula:	Turma

Considere a existência de um vetor de `Aluno` ordenado pelo critério da questão anterior, isto é, os alunos estão ordenados por turma, usando o nome em ordem alfabética como critério de desempate.

Sem usar funções da biblioteca padrão, escreva uma função que faça uma *busca binária* no vetor. Essa função recebe como parâmetros de entrada o número de alunos, o vetor, a turma e o nome do aluno que se deseja buscar. A função deve ter como valor de retorno o índice no vetor da estrutura Aluno com turma e nome iguais aos fornecidos. Se não existir um aluno com os dados fornecidos, a função deve ter `-1` como valor de retorno. O protótipo da função deve ser dado por:

```
int busca (int n, Aluno* vet, char turma, char* nome);
```

```
/* uma alternativa é usar uma função auxiliar p/ comparar */
int comp_aux(char t, char* n, Aluno* a)
{
   if (t < a->turma)
      return -1;
   else if (t > a->tuma)
      return 1;
   else
      return strcmp(n,a->nome);
}

/* função de busca (alternativa de não usar recursão) */
int busca (int n, Aluno* vet, char turma, char* nome)
{
   int i=0;
   int f=n-1;
   while (i<=f) {
      int m = (i+f)/2;
      int r = comp_aux (turma,nome,&vet[m]);
      if (r < 0)
         f = m-1;
      else if (r > 0)
         i = m+1;
      else
         return m;
   }
   return -1;
}
```

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 – P3 - 02/07/02	Questão 4
Nome:	
Matrícula:	Turma

Considere, novamente, a existência de um vetor de `Aluno` ordenado pelo critério da questão anterior, isto é, os alunos estão ordenados por turma, usando o nome em ordem alfabética como critério de desempate.

Escreva uma função que salve num arquivo texto a relação dos alunos por turma. O formato do arquivo de saída deve apresentar numa linha a turma, seguido da lista de alunos da turma em questão. Entre turmas distintas, deve-se prever uma linha em branco. Um exemplo deste formato é mostrado a seguir.

```
TURMA A
Fulano de Tal
Sicrano Silva

TURMA B
Beltrano Alves
Maria de Souza
```

A função deve receber o nome do arquivo de saída, o número de alunos e o vetor. Se ocorrer erro na abertura do arquivo, a função deve retornar `0`, caso contrário o conteúdo do vetor deve ser salvo segundo o formato acima e a função deve retornar `1`. O protótipo da função é dado por:

```
int salva (char* arquivo, int n, Aluno* vet);
```

```
int salva (char* arquivo, int n, Aluno* vet)
{
   int i;
   char t = 0;    /* turma corrente */
   FILE* fp = fopen(arquivo,"wt");

   if (fp==NULL)
      return 0;

   for (i=0; i<n; i++) {
      if (vet[i].turma != t) {
         if (t!=0)  /* primeira linha ja' foi escrita */
            fprintf(fp,"\n");
         t = vet[i].turma;
         fprintf(fp,"TURMA %c\n", t);
      }
      fprintf(fp,"%s\n",vet[i].nome);
   }
   fclose(fp);
   return 1;
}
```

Protótipos de funções que podem ser úteis:

```
FILE* fopen (char* nome, char* modo);
int fclose (FILE* fp);
int fprintf (FILE* fp, char* formato, ...);
```

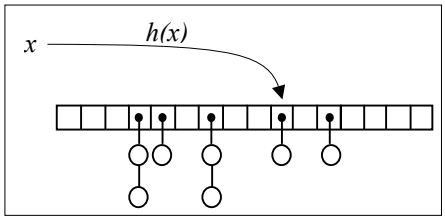
Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.

INF 1620 – P3 - 02/07/02	Questão 5
Nome:	
Matrícula:	Turma

Considere um cadastro de produtos armazenado numa tabela de dispersão (*hash*) que usa como chave de busca o nome do produto. Considere também a existência de uma função de *hash* que, dado o nome do produto, retorna o índice na tabela. O protótipo dessa função é dado por:

```
int hash (char* nome);
```

O tratamento de colisão é feito utilizando-se listas encadeadas para armazenar os elementos que colidem. Desta forma, cada posição na tabela representa um ponteiro para o primeiro nó de uma lista encadeada, conforme ilustra a figura ao lado.



Considere as seguintes declarações, relativas à tabela de *hash* dos produtos.

```
struct produto {
   char nome[81];            /* nome do produto */
   int quant;                /* quantidade em estoque */
   float preço;              /* preço do produto */
   struct produto* prox;     /* próximo elemento da lista*/
};
typedef struct produto Produto;

#define N 127
typedef Produto* Hash[N];
```

Considerando que posições não preenchidas da tabela de *hash* armazenam o valor `NULL` (lista vazia), escreva uma função que, dado o nome de um produto, tenha como valor de retorno a quantidade disponível no estoque. Se o produto não estiver presente no estoque, a função deve ter `-1` como valor de retorno. O protótipo da função deve ser:

```
int busca (Hash tab, char* nome);
```

```
int busca (Hash tab, char* nome)
{
   Produto* p;
   int h = hash(nome);

   for (p=tab[h]; p!=NULL; p=p->prox) {
      if (strcmp(nome,p->nome) == 0)
         return p->quant;
   }
   return -1;
}
```

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