INF 1620 – P3 - 27/11/04	Questão 1
Nome:	
Matrícula:	Turma

Considere um arquivo texto com as notas dos alunos de uma disciplina. Os dados de cada aluno são armazenados em duas linhas do arquivo: uma com o seu nome (cadeia com até 80 caracteres), e outra com suas três notas (P1, P2 e P3). Considere ainda que não existem linhas em branco no arquivo. Um exemplo desse formato é mostrado abaixo.

```
João da Silva
2.0 4.3 6.5
Manuel Santos
7.0 8.2 8.6
Fulana de Tal
6.0 7.5 7.8
```

Escreva um programa completo que leia o arquivo "entrada.txt", que contém as notas de uma disciplina no formato descrito acima, e escreva no arquivo texto "saida.txt" o nome de cada aluno com sua respectiva média ((P1+P2+P3)/3) e um caractere indicando se o aluno foi aprovado ou reprovado na disciplina (use 'A' para indicar aprovação e 'R' para indicar reprovação). Considere que um aluno está aprovado se tiver média maior ou igual a 5.0. Para o exemplo acima, o programa deve criar um arquivo "saida.txt" com o seguinte conteúdo:

```
João da Silva 4.3 R
Manuel Santos 7.9 A
Fulana de Tal 7.1 A
```

Observe que as médias são escritas no arquivo "saida.txt" com apenas uma casa decimal. Se não for possível abrir um dos arquivos, o programa deve imprimir a mensagem "ERRO" e terminar sua execução

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 - 27/11/04	Questão 2
Nome:	
Matrícula:	Turma

Considere um tipo que representa um registro de uma agenda de telefones (com nome, telefone e data da nascimento) definido pela estrutura a seguir:

```
struct contato {
   char nome[81];
   char telefone[21];
   int dia;
   int mes;
   int ano;
};
typedef struct contato Contato;
```

Usando alguma das técnicas de ordenação vistas no curso, escreva uma função para ordenar um vetor de contatos, em ordem crescente de data de nascimento. A função deve receber como parâmetros o número de contatos e um vetor que armazena ponteiros para estruturas do tipo `Contato`, de acordo com o protótipo definido a seguir:

```
void ordena (int n, Contato** v);
```

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 - 27/11/04	Questão 3
Nome:	
Matrícula:	Turma

Considere um vetor de números inteiros ordenado em ordem crescente, onde não existem valores repetidos. Utilizando a técnica de *busca binária* em vetores, implemente uma função que determine o índice em que um novo elemento deve ser inserido no vetor de tal forma que a ordenação do vetor seja preservada (assuma que o novo elemento é diferente dos demais). Por exemplo, para o vetor {1,3,6,8,9,12} e um novo elemento com valor 5, sua função deve retornar o índice 2. Sua função recebe como parâmetros um número inteiro `n` com a quantidade de elementos correntemente no vetor, o vetor de inteiros `v` e o número inteiro `x` a ser inserido no vetor, e retorna o índice da posição do novo elemento, de acordo com o cabeçalho a seguir:

```
int indice (int n, int* v, int x);
```

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 - 27/11/04	Questão 4
Nome:	
Matrícula:	Turma

Considere uma *árvore binária de busca* que armazena valores inteiros, onde os valores associados aos nós da sub-árvore à esquerda são menores que o valor associado à raiz e que os valores dos nós da sub-árvore à direita são maiores (considere que a árvore não possui valores repetidos). O tipo que representa um nó da arvore é dado por:

```
struct arv {
   int val;
   struct arv* esq;
   struct arv* dir;
};
typedef struct arv Arv;
```

Escreva uma função que retorne o número de nós da árvore cuja informação armazenada seja menor que um dado valor `x`. A função deve tirar proveito da ordenação da árvore e obedecer ao seguinte protótipo:

```
int menores_x (Arv* a, int x);
```

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 - 27/11/04	Questão 5
Nome:	
Matrícula:	Turma

Considere um cadastro de alunos armazenado em uma tabela de dispersão (*hash*) que usa listas encadeadas para o tratamento de colisões, isto é, a tabela é implementada como um vetor de listas encadeadas. Considerando as seguintes declarações relativas à tabela *hash* dos alunos:

```
struct aluno {
   char nome[81];
   int matricula;
   float cr;
   struct aluno* prox;
};
typedef struct aluno Aluno;

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

Implemente:

a) Uma função de *hash* que, dado o nome de um aluno, retorne o índice do aluno na tabela, de acordo com protótipo definido a seguir:

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

b) Uma função que insira um novo aluno na tabela *hash*. Para simplificar, considere que a função estará sempre adicionando um novo aluno ao cadastro, isto é, não existe a hipótese do nome do aluno a ser inserido já constar no cadastro. A função deve receber como parâmetros a tabela *hash*, o nome do aluno, sua matrícula e seu CR, e deve retornar um ponteiro para a estrutura criada para o novo aluno, de acordo com o protótipo definido a seguir:

```
Aluno* insere(Hash tab, char* nome, int mat, float cr);
```

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.

Protótipos de funções que podem ser úteis:

stdio.h:

```
int scanf (char* formato, ...);
int printf (char* formato, ...);
FILE* fopen (char* nome, char* modo);
int fclose (FILE* fp);
int fscanf (FILE* fp, char* formato, ...);
int fprintf (FILE* fp, char* formato, ...);
char* fgets(char* str, int size, FILE* fp));
int sscanf(char* str, char* formato, ...);
```

math.h:

```
double sqrt (double x);
double pow (double x, double exp);
```

string.h:

```
int strlen (char* s);
int strcmp (char* s, char *t);
char* strcpy (char* destino, char* fonte);
char* strcat (char* destino, char* fonte);
```

stdlib.h:

```
void* malloc (int nbytes);
void free (void* p);
void qsort (void *vet, int n, int tam, int (*comp) (const void*, const void*));
```

Não separe as folhas deste caderno.