Lista de Exercícios

Leitura de Código

Questão 01 [1xp] - qual a saída do programa abaixo?

```c
#include <stdio.h>

struct DoisValores
{
   int vi;
   float vf;
};

int main( void )
{
   struct DoisValores reg1 = { 53, 7.112 }, reg2, *p = &reg1;
   reg2.vi = (*p).vf;
   reg2.vf = (*p).vi;

   printf( "1: %d %f\n", reg1.vi, reg1.vf );
   printf( "2: %d %f\n", reg2.vi, reg2.vf );
   return 0;
}
```

Questão 02 [3xp] - qual a saída do programa abaixo para os valores (4, 2), (7, 5) ?

```c
#include <stdio.h>

struct Celula
{
   int valor;
   struct Celula *proximo;
};

int main( void )
{
   struct Celula reg1, reg2, *p;
   scanf( "%d %d", &reg1.valor, &reg2.valor );
   reg1.proximo = &reg2;
   reg2.proximo = NULL;
   for( p = &reg1; p != NULL; p = p->proximo )
       printf( "%d ", p->valor );
   printf( "\n" );
   return 0;
}
```

Questão 03 [2xp] - qual a saída do programa abaixo para os valor: 4 e 7?

```c
#include <stdio.h>
#define MAX 50
int main() {
    int n;
    int i, j;
    int mat[MAX][MAX];

    while(scanf("%d", &n) != EOF){
        for(i=0; i<n; i++){
```

```
            for(j=0; j<n; j++){
                if(i+j == n-1)
                    mat[i][j] = 2;
                else if(i == j)
                    mat[i][j] = 1;
                else
                    mat[i][j] = 3;
            }
         }
    }

    for(i=0; i<n; i++){
       for(j=0; j<n; j++)
           printf("%d ", mat[i][j]);
       printf("\n");

    }

    return 0;
}
```

Ache o erro

Questão 01 [4xp] - o programa abaixo foi feito para validar se uma dada matriz é uma matriz Identidade. Porém ele falha para alguns casos. Identifique o(s) erro(s):

```
#include <stdio.h>
#define N 10

int checa_matriz(int mat[N][N]){
    int i, j;

    for(i=1; i<=N; i++){
        for(j=1; j<=N; j++){
            if(i == j){
             if(mat[i][j] == 1)
                   return 0;
            }
            else{
             if(mat[i][j] == 0)
                   return 0;
            }
        }
    }
    return 1;
}

int main(){
    int mat[N][N] = {0};
    int ok;
    int i, j;

    for(i=1; i<=N; i++)
       for(j=1; j<=N; j++)
            scanf("%d", mat);
```

```
    ok = checa_matriz(mat);

    if(ok == 1)
       printf("Eh matriz identidade!\n");
    else
       printf("Nao eh matriz identidade!\n");

    return 0;
}
```

Complete o código

Questão 01 [5xp] - complete o código com as subrotinas não-declaradas, como descritas nos comentários do código.

```
#include <stdio.h>

typedef struct {
   int matricula;
   float notas[3];
} Aluno;

int main( void )
{
   Aluno turma[5];
   int i;
   for( i = 0; i < 5; i++ )
       le_aluno( &turma[i] );   /* lê os dados do aluno da entrada padrão */

   imprime_turma( turma, 5 );   /* imprime os dados de todos os alunos */
   for( i = 0; i < 5; i++ )
   {
         float media = calcula_media( turma[i].notas, 3 ); /* calcula a média das
notas */
       printf( "Aluno %i: Media = %f\n", i + 1, media );
   }
   return 0;
}
```

Problemas para Resolver

Questão 01 [2xp] - (Kodesh)

Uma grade do jogo Sudoku clássico é um quadrado latino 9×9 com restrições adicionais: cada uma das 9 sub-divisões de quadrados 3×3 deve conter números distintos entre 1 e 9, ou seja, um Sudoku é um quadrado latino 9×9 que contém 9 quadrados 3×3 com números que não se repetem.

No jogo de Sudoku, apenas algumas posições da grade são preenchidas e o jogador deve preencher as demais posições de forma a satisfazer as regras de disposição.

Escreva uma sub-rotina que testa se uma matriz de inteiros 9×9 satisfaz as condições de uma grade de Sudoku preenchida. Sua sub-rotina vai receber uma matriz de inteiros e deve retornar 1 (um) caso seja um jogo válido de Sudoku e 0 (zero) caso contrário. Observe a interface da função:

```
int                 eh_sudoku                 (int                 sud[9][9]);
```

Um programa receberá da entrada padrão uma matriz e utilizará sua função para verificar se a grade lida de Sudoku é uma jogada válida.

Entrada	**Saída**
9 4 8 1 6 7 2 5 3 5 6 3 9 2 4 8 1 7 2 7 1 3 8 5 4 6 9 1 5 4 7 3 2 6 9 8 7 9 2 8 1 6 5 3 4 3 8 6 4 5 9 7 2 1 4 2 9 5 7 1 3 8 6 6 3 7 2 9 8 1 4 5 8 1 5 6 4 3 9 7 2	S
1 9 8 7 6 5 4 3 2 4 3 2 1 9 8 7 6 5 7 6 5 4 3 2 1 9 8 2 1 9 8 7 6 5 4 3 9 8 7 6 5 4 3 2 1 5 4 3 2 1 9 8 7 6 6 5 4 3 2 1 9 8 7 3 2 1 9 8 7 6 5 4 8 7 6 5 4 3 2 1 9	N
7 9 6 8 5 3 2 1 4 5 2 4 9 6 1 3 8 7 1 3 8 2 7 4 5 6 9 2 1 9 6 8 7 4 3 5 8 4 5 3 2 9 1 7 6 6 7 3 4 1 5 9 2 8 3 5 2 7 4 8 6 9 1 4 6 7 1 9 2 8 5 3 9 8 1 5 3 6 7 4 2	S
1 9 8 7 6 5 4 3 2 2 1 9 8 7 6 5 4 3 3 2 1 9 8 7 6 5 4 4 3 2 1 9 8 7 6 5 5 4 3 2 1 9 8 7 6 6 5 4 3 2 1 9 8 7 7 6 5 4 3 2 1 9 8	N

8 7 6 5 4 3 2 1 9
9 8 7 6 5 4 3 2 1

Questão 02 [4xp]
Escreva um programa que armazena o seguinte conjunto de informações de um funcionário:

NOME	ENDEREÇO	SALÁRIO
IDENTIDADE	CPF	ESTADO CIVIL
TELEFONE	IDADE	SEXO

O ENDEREÇO é composto de:

RUA	BAIRRO	CIDADE
ESTADO	CEP	

Seu programa deve armazenar um conjunto de 10 funcionários, com informações preenchidas através da entrada padrão. Posteriormente, seu programa deve imprimir o nome dos funcionários cujo salário >= 2000.00, idade entre 20 e 35 anos e vem do Estado do RN.

Questão 03 [5xp]

Sempre em reuniões internacionais, professor Ivonildo precisa de ajuda para organizar os seu dia. Para isso ele pediu que os alunos façam um programa de agenda particular, IMDiary. O IMDiary deve armazenar as informações de compromissos com os seguintes dados:

- Horário de início
- Horário de fim
- Descrição da atividade
- Lista de pessoas com quem ele desempenha a atividade (pode ser vazio)

Seu programa deve iniciar lendo um número N (1 <= N <= 100) contendo o número de tarefas que o professor colocará na agenda, seguido de n pares de linhas no seguinte formato:

Linha1 - hh1:mm1 hh2:mm2 (Pessoa1, Pessoa2, …, PessoaN)
Linha2 - Texto_descrição

onde

hh1:mm1 é a hora de início, de 00:00 até 23:59

hh2:mm2 é a hora final, de 00:00 até 23:59

Pessoa1, Pessoa2 …. PessoaN são nomes de pessoas, sem espaços em branco (nomes simples), podendo ser uma lista vazia. Nesse caso aparecerá () na linha de entrada. Cada nome não será maior do que 15 caracteres.

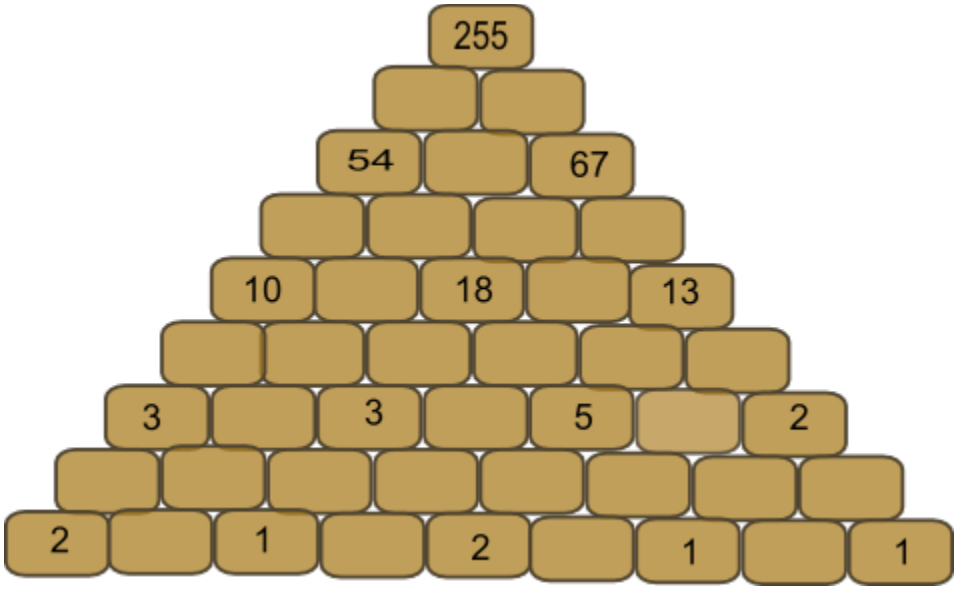
Texto_descrição é uma string contendo uma breve descrição da atividade, podendo haver espaço em branco no texto. Não será maior do que 100 caracteres.

Após a leitura das informações, seu programa lerá um número indefinido de linhas contendo dois horários no formato hh:mm, indicando um intervalo de tempo que o professor deseja consultar. Para esse intervalo, o seu programa deve imprimir uma lista das atividades registradas na agenda que acontecem nesse horário, em ordem cronológica. Uma linha em branco deve ser impressa separando as saídas de intervalos diferentes, conforme exemplo abaixo.

Entrada	**Saída**
5 08:00 09:00 () Café da manhã 10:00 12:00 (Alyson) Partida de tênis 14:00 17:00 (André, Charles, Bruno, Rafaela) Reunião Estágio Probatório 19:30 22:00 () Mad Max no Cinépolis 23:30 23:59 (Cérebro, Pinky) Dominação Mundial 09:30 15:00 19:30 23:45	10:00 - Partida de Tênis 14:00 - Reunião Estágio Probatório 19:30 - Mad Max no Cinépolis 23:30 - Dominação Mundial

Questão 04 (URI 1426) [6xp]

Não, não é "mais um tijolo na parede", é apenas um problema sobre somar números.
Suponha que você tem uma parede com o formato de um triângulo, como a mostrada abaixo. A parede tem 9 linhas, e a i-ésima linha tem exatamente i tijolos, considerando que a linha mais acima é a 1ª e que a mais abaixo é a 9ª. Alguns tijolos são rotulados com um número, enquanto os demais estão em branco. Os tijolos rotulados aparecem apenas em linhas ímpares, e ocupam posições ímpares dentro das suas linhas.



O problema que você deve resolver consiste em rotular os tijolos em branco com números, de tal forma que a seguinte regra seja satisfeita: O número de um tijolo é igual à soma dos números dos dois tijolos abaixo dele. Obviamente, esta regra não é aplicada à 9ª linha. Todos os números devem ser inteiros.
Nota: O exemplo de entrada contém dois casos de teste. O primeiro dele corresponde à parede mostrada acima.

A primeira linha da entrada contém um inteiro **N**, indicando o número de casos de teste. Esta linha é seguida pelos casos de teste. Cada caso é descrito por 5 linhas. Essas linhas correspondem às linhas ímpares da parede, de cima para baixo, como descrito acima. Cada linha contém os números nos tijolos já rotulados da linha correspondente na parede, da esquerda para a direita, separados por um espaço em branco.
Você pode assumir que todo caso de teste é correto, isto é, existe uma solução para o problema descrito.

Para cada caso de teste, a saída deve ser formada por 9 linhas descrevendo os números de todos os tijolos da parede. Assim, a **i**-ésima linha deve conter os números correspondentes aos **i** tijolos da **i**-ésima linha da parede, da esquerda para a direita, separados por um espaço.

Entrada

```
2
255
54 67
10 18 13
3 3 5 2
2 1 2 1 1

256
64 64
16 16 16
4 4 4 4
```

Saída

```
255
121 134
54 67 67
23 31 36 31
10 13 18 18 13
5 5 8 10 8 5
3 2 3 5 5 3 2
2 1 1 2 3 2 1 1
2 0 1 0 2 1 1 0 1

256
```

1 1 1 1 1

128 128
64 64 64
32 32 32 32
16 16 16 16 16
8 8 8 8 8 8
4 4 4 4 4 4 4
2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1