21111 - Sistemas Operativos: proposta de resolução do eFólio B

E-fólio B, notas lançadas
por José Coelho - Terça, 24 Maio 2022, 09:21

Número de respostas: 12

Caros estudantes,

Estão lançadas as notas do e-fólio B. Temos 60 estudantes que entregaram, tendo a distribuição das notas pelos estudantes a seguinte:

- 5 valores: 1 estudante
- [4; 5[valores: 16 estudantes
- [3; 4[valores: 11 estudantes
- [2; 3[valores: 16 estudantes
- [1; 2[valores: 11 estudantes
- [0; 1[valores: 5 estudantes

Em termos de nota na avaliação contínua, a distribuição das notas foi a seguinte:

- [7,5; 8] valores: 9 estudantes
- {6,5; 7,5[valores: 21 estudantes
- {5,5; 6,5[valores: 11 estudantes
- {4,5; 5,5[valores: 6 estudantes
- {3,5; 4,5[valores: 5 estudantes
- {2,5; 3,5[valores: 3 estudantes
- {1,5; 2,5[valores: 4 estudantes
- {0,5; 1,5[valores: 7 estudantes
- {0; 0,5[valores: 2 estudantes

Os estudantes com mais de 3,5 valores devem ir na época normal ao e-fólio global (52 estudantes), quem ficou a baixo dos 3,5 valores ou não entregou os e-fólios e esteve em avaliação contínua, pode ir a exame mas apenas na época de recurso.

Temos 30 estudantes com mais de 6,5 valores, pelo que é um bom registo e vão a caminho de uma excelente nota, e é importante que todos os outros se possam preparar para obterem aprovação com uma boa nota, de preferência já na época normal.

Anexo duas possíveis resoluções do e-fólio B, com e sem alocação de memória dinâmica.

Qualquer questão que tenham sobre a correção, sabem que podem perguntar e clarificar com o avaliador, tal como no e-fólio A.

Cumprimentos,
José Coelho

Anexos: pteste.c, pteste2.c e casos.txt

```c
// pteste.c
// Proposta de resolução do eFólio-B 2022
// sem alocação de memória dinâmica

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

#define MAX_STR 255
#define MAX_CASES 1000
#define MAXTRF 30       /* numero maximo de tarefas */

void* tarefa(void* arg);

int casos_teste[MAX_CASES][2];   // argumentos de cada caso de teste
int resultados[MAX_CASES][2]; // resultados esperados e observados
int totalCasos, sucessos, falhas, casosProcessados; // vari�veis de controlo e finais
pthread_mutex_t mtx_casos, mtx_sucessos, mtx_falhas; /* mutexes */


int Combinacoes(int n, int r)
{
    int i, combinacoes;

    if (n < r || r < 1)
        return 0;

    i = 1;
    combinacoes = 1;
    while (i <= r)
    {
        combinacoes *= (n - r + i);
        combinacoes /= i;
        i++;
    }
    return combinacoes;
}

int LerCasos(char *casos) {
    FILE* f;
    char str[MAX_STR], *pt;
    casosProcessados = 0;
    totalCasos = 0;
    sucessos = 0;
    falhas = 0;
    if ((f = fopen(casos, "rt")) != NULL) {
        while (fgets(str, MAX_STR, f) != NULL) {
            pt = strtok(str, " ");
            casos_teste[totalCasos][0] = atoi(pt);
            pt = strtok(NULL, " ");
            casos_teste[totalCasos][1] = atoi(pt);
```

```
            pt = strtok(NULL, " ");
            resultados[totalCasos][0] = atoi(pt);
            totalCasos++;
            if (totalCasos >= MAX_CASES - 1)
                break;
        }
        fclose(f);
        return 1;
    }
    printf("\nErro ao ler o ficheiro de casos.");
    return 0;
}


int main(int argc, char **argv)
{
    int i, r, N;
    pthread_t      trfid[MAXTRF];    /* variavel para ID das tarefas */
    pthread_attr_t trfatr;           /* variavel para atributos das tarefas */

    N = atoi(argv[1]);
    if (N < 1)
        N = 1;

    // ler dados
    if (!LerCasos(argv[2])) // atualiza asvari□veis globais
        exit(1);

    /* inicializar variavel de atributos com valores por defeito */
    pthread_attr_init(&trfatr);
    /* modificar estado de desacoplamento para "joinable" */
    pthread_attr_setdetachstate(&trfatr, PTHREAD_CREATE_JOINABLE);
    /* inicializar mutex com valores por defeito para os atributos */
    pthread_mutex_init(&mtx_casos, NULL);
    pthread_mutex_init(&mtx_sucessos, NULL);
    pthread_mutex_init(&mtx_falhas, NULL);

    /* inicializar estruturas de dados que servem de argumento das tarefas */
    for (i = 0; i < N; i++) {
        /* criar e iniciar execucao de tarefa */
        r = pthread_create(&trfid[i], &trfatr, tarefa, NULL);
        if (r) {
            /* erro ! */
            perror("Erro na criacao da tarefa!");
            exit(1);
        }
    }

    /* esperar que as tarefas criadas terminem */
    for (i = 0; i < N; i++)
        pthread_join(trfid[i], (void**)NULL);
    /* libertar recursos associados ao mutex */
```

```
    pthread_mutex_destroy(&mtx_casos);
    pthread_mutex_destroy(&mtx_sucessos);
    pthread_mutex_destroy(&mtx_falhas);

    printf("\nResultados: \nSucesso: %d\nFalhas: %d\nTotal casos: %d.",
        sucessos, falhas, totalCasos);

    for (i = 0; i < totalCasos; i++)
        if (resultados[i][0] != resultados[i][1])
            printf("\nErro no caso %d: input (%d, %d) esperado/observado: %d / %d.",
                i, casos_teste[i][0], casos_teste[i][1],
                resultados[i][0], resultados[i][1]);

    return 0;
}

void* tarefa(void* arg)
{
    int item=-1;
    do {
        // obter um valor para processar
        pthread_mutex_lock(&mtx_casos);
        item = casosProcessados++;
        pthread_mutex_unlock(&mtx_casos);
        if (item < totalCasos) {
            // executa e grava o resultado observado
            if ((resultados[item][1]=Combinacoes(casos_teste[item][0],
casos_teste[item][1]))
                == resultados[item][0])
            {
                // sucesso
                pthread_mutex_lock(&mtx_sucessos);
                sucessos++;
                pthread_mutex_unlock(&mtx_sucessos);
            }
            else {
                // insucesso
                pthread_mutex_lock(&mtx_falhas);
                falhas++;
                pthread_mutex_unlock(&mtx_falhas);
            }
        }
    } while (item<totalCasos);
    return (void*)NULL;
}
```

```c
// pteste2.c
// Proposta de resolução do eFólio-B 2022
// com alocação de memória dinâmica

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

#define MAX_STR 255
#define MAX_CASES 1000

void* tarefa(void* arg);

typedef struct {
    int *casos_teste[2];   // argumentos de cada caso de teste
    int *resultados[2]; // resultados esperados e observados
    int totalCasos, sucessos, falhas, casosProcessados; // vari□veis de controlo e
finais
    pthread_mutex_t mtx_casos, mtx_sucessos, mtx_falhas; /* mutexes */
} TMemTarefas;

// chamar ap□s totalCasos estar preenchido
void AlocarMemTarefas(TMemTarefas* mem)
{
    int i;
    for (i = 0; i < 2; i++) {
        mem->casos_teste[i] = (int*)malloc(mem->totalCasos * sizeof(int));
        mem->resultados[i] = (int*)malloc(mem->totalCasos * sizeof(int));
        if (mem->casos_teste[i] == NULL ||
            mem->resultados[i] == NULL)
            exit(1);
    }
}

void LibertarMemTarefas(TMemTarefas* mem)
{
    int i;
    for (i = 0; i < 2; i++) {
        free(mem->casos_teste[i]);
        free(mem->resultados[i]);
    }
}


int Combinacoes(int n, int r)
{
    int i, combinacoes;

    if (n < r || r < 1)
        return 0;
```

```
    i = 1;
    combinacoes = 1;
    while (i <= r)
    {
        combinacoes *= (n - r + i);
        combinacoes /= i;
        i++;
    }
    return combinacoes;
}

int LerCasos(TMemTarefas*mem, char *casos) {
    FILE* f;
    char str[MAX_STR], *pt;
    int i;
    mem->casosProcessados = 0;
    mem->totalCasos = 0;
    mem->sucessos = 0;
    mem->falhas = 0;
    if ((f = fopen(casos, "rt")) != NULL) {
        // duas passagens pelo ficheiro, primeira passagem para contar as linhas
        while (fgets(str, MAX_STR, f) != NULL)
            mem->totalCasos++;
        // alocar mem�ria
        AlocarMemTarefas(mem);

        fseek(f, 0, SEEK_SET); // retomar para o in�cio
        i = 0;
        while (fgets(str, MAX_STR, f) != NULL) {
            pt = strtok(str, " ");
            mem->casos_teste[0][i] = atoi(pt);
            pt = strtok(NULL, " ");
            mem->casos_teste[1][i] = atoi(pt);
            pt = strtok(NULL, " ");
            mem->resultados[0][i] = atoi(pt);
            i++;
        }
        fclose(f);
        return 1;
    }
    printf("\nErro ao ler o ficheiro de casos.");
    return 0;
}



int main(int argc, char **argv)
{
    int i, r, N;
    TMemTarefas mem;
    pthread_t      *trfid;           /* variavel para ID das tarefas */
    pthread_attr_t trfatr;           /* variavel para atributos das tarefas */
```

```
    N = atoi(argv[1]);
    if (N < 1)
        N = 1;

    // ler dados
    if (!LerCasos(&mem, argv[2])) // atualiza asvari□veis globais
        exit(1);

    trfid = (pthread_t*)malloc(N * sizeof(pthread_t));
    if (trfid == NULL)
        exit(1);

    /* inicializar variavel de atributos com valores por defeito */
    pthread_attr_init(&trfatr);
    /* modificar estado de desacoplamento para "joinable" */
    pthread_attr_setdetachstate(&trfatr, PTHREAD_CREATE_JOINABLE);
    /* inicializar mutex com valores por defeito para os atributos */
    pthread_mutex_init(&mem.mtx_casos, NULL);
    pthread_mutex_init(&mem.mtx_sucessos, NULL);
    pthread_mutex_init(&mem.mtx_falhas, NULL);

    /* inicializar estruturas de dados que servem de argumento das tarefas */
    for (i = 0; i < N; i++) {
        /* criar e iniciar execucao de tarefa */
        r = pthread_create(&trfid[i], &trfatr, tarefa, (void*) & mem);
        if (r) {
            /* erro ! */
            perror("Erro na criacao da tarefa!");
            exit(1);
        }
    }

    /* esperar que as tarefas criadas terminem */
    for (i = 0; i < N; i++)
        pthread_join(trfid[i], (void**)NULL);
    /* libertar recursos associados ao mutex */
    pthread_mutex_destroy(&mem.mtx_casos);
    pthread_mutex_destroy(&mem.mtx_sucessos);
    pthread_mutex_destroy(&mem.mtx_falhas);

    printf("\nResultados: \nSucesso: %d\nFalhas: %d\nTotal casos: %d.",
        mem.sucessos, mem.falhas, mem.totalCasos);

    for (i = 0; i < mem.totalCasos; i++)
        if (mem.resultados[0][i] != mem.resultados[1][i])
            printf("\nErro no caso %d: input (%d, %d) esperado/observado: %d / %d.",
                i, mem.casos_teste[0][i], mem.casos_teste[1][i],
                mem.resultados[0][i], mem.resultados[1][i]);

    free(trfid);
    LibertarMemTarefas(&mem);
```

```
    return 0;
}

void* tarefa(void* arg)
{
    int item = -1;
    TMemTarefas* mem = (TMemTarefas*)arg;
    do {
        // obter um valor para processar
        pthread_mutex_lock(&mem->mtx_casos);
        item = mem->casosProcessados++;
        pthread_mutex_unlock(&mem->mtx_casos);
        if (item < mem->totalCasos) {
            // executa e grava o resultado observado
            if ((mem->resultados[1][item] = Combinacoes(mem->casos_teste[0][item],
mem->casos_teste[1][item]))
                == mem->resultados[0][item])
            {
                // sucesso
                pthread_mutex_lock(&mem->mtx_sucessos);
                mem->sucessos++;
                pthread_mutex_unlock(&mem->mtx_sucessos);
            }
            else {
                // insucesso
                pthread_mutex_lock(&mem->mtx_falhas);
                mem->falhas++;
                pthread_mutex_unlock(&mem->mtx_falhas);
            }
        }
    } while (item < mem->totalCasos);
    return (void*)NULL;
}
```

casos.txt (998 casos de teste: 1 por linha – N,R,Combinação)

8 4 70
1 1 1
2 2 1
4 4 1
6 5 6
3 1 3
6 4 15
6 6 1
1 1 1
10 1 10
1 1 1
9 4 126
1 1 1
10 9 10
8 8 1
8 4 70
1 1 1
10 10 1
2 2 1
6 4 15
3 2 3
7 5 21
5 4 5
4 1 4
5 2 10
7 5 21
1 1 1
3 3 1
4 1 4
10 5 252
1 1 1
1 1 1
10 9 10
11 11 1
8 5 56
4 2 6
1 1 1
1 1 1
10 3 120
1 1 1
5 1 5
9 3 84
2 2 1
6 3 20
11 8 165
9 2 36
5 5 1
1 1 1
12 5 792
4 3 4
10 8 45
3 1 3
2 2 1

10 6 210
12 2 66
3 3 1
4 4 1
5 1 5
11 3 165
5 2 10
8 6 28
7 3 35
10 9 10
6 1 6
8 6 28
9 9 1
7 2 21
12 9 220
11 1 11
2 2 1
9 5 126
12 6 924
5 1 5
7 6 7
2 1 2
9 1 9
12 6 924
13 13 1
1 1 1
7 6 7
10 7 120
10 5 252
6 6 1
3 2 3
2 1 2
14 2 91
8 3 56
6 4 15
2 2 1
2 1 2
2 2 1
7 7 1
14 1 14
8 5 56
11 8 165
13 4 715
12 11 12
11 10 11
3 1 3
13 8 1287
6 5 6
15 6 5005
3 2 3
1 1 1
9 1 9
3 1 3
6 4 15
2 1 2

7 2 21
4 4 1
1 1 1
3 1 3
12 2 66
6 1 6
14 5 2002
3 2 3
11 7 330
11 5 462
15 11 1365
14 9 2002
8 7 8
16 9 11440
6 2 15
7 7 1
3 3 1
11 2 55
3 1 3
10 5 252
15 15 1
16 3 560
6 6 1
8 6 28
13 1 13
7 4 35
5 5 1
13 6 1716
11 9 55
8 2 28
1 1 1
3 1 3
10 10 1
8 6 28
2 2 1
16 14 120
13 8 1287
8 3 56
10 2 45
9 3 84
17 6 12376
16 3 560
8 8 1
11 2 55
5 5 1
13 2 78
10 7 120
12 5 792
3 3 1
12 10 66
13 8 1287
9 3 84
9 5 126
16 9 11440
13 9 715

1 1 1
6 4 15
3 2 3
6 1 6
10 9 10
18 11 31824
4 4 1
3 1 3
7 1 7
9 4 126
10 7 120
17 12 6188
6 6 1
5 2 10
18 13 8568
5 4 5
17 4 2380
10 7 120
6 1 6
17 11 12376
2 2 1
13 7 1716
12 6 924
13 1 13
16 9 11440
2 1 2
18 14 3060
6 6 1
11 8 165
15 11 1365
15 6 5005
19 1 19
18 13 8568
5 5 1
7 2 21
10 8 45
15 12 455
10 7 120
17 14 680
9 7 36
2 1 2
3 1 3
5 3 10
6 1 6
14 13 14
11 11 1
12 4 495
8 2 28
5 4 5
3 1 3
8 3 56
10 9 10
5 4 5
16 14 120
15 11 1365

20 4 4845
7 6 7
15 7 6435
16 13 560
17 10 19448
8 5 56
4 3 4
20 17 1140
18 12 18564
2 1 2
16 10 8008
18 1 18
4 3 4
18 7 31824
10 8 45
16 7 11440
21 8 203490
2 1 2
13 9 715
13 7 1716
5 3 10
15 12 455
1 1 1
12 1 12
10 7 120
14 9 2002
19 19 1
13 1 13
6 4 15
8 2 28
5 1 5
6 4 15
17 13 2380
15 11 1365
5 1 5
7 4 35
18 5 8568
5 5 1
2 2 1
18 12 18564
9 1 9
5 5 1
4 2 6
17 1 17
8 7 8
16 14 120
1 1 1
22 17 26334
5 2 10
6 3 20
9 2 36
4 2 6
17 14 680
3 3 1
4 3 4

10 6 210
8 3 56
21 4 5985
22 9 497420
9 9 1
23 11 1352078
20 17 1140
8 8 1
16 12 1820
5 3 10
5 3 10
16 9 11440
8 1 8
16 10 8008
5 3 10
5 3 10
15 10 3003
1 1 1
1 1 1
15 11 1365
13 5 1287
10 2 45
2 2 1
4 3 4
17 7 19448
5 2 10
18 4 3060
14 9 2002
8 1 8
15 13 105
6 3 20
7 5 21
3 2 3
20 3 1140
18 14 3060
17 6 12376
10 4 210
9 4 126
7 6 7
14 5 2002
9 3 84
18 5 8568
17 17 1
17 7 19448
8 5 56
18 15 816
3 3 1
3 1 3
9 9 1
1 1 1
18 14 3060
22 1 22
21 5 20349
20 4 4845
5 2 10

12 3 220
26 15 7726160
15 7 6435
13 3 286
25 18 480700
8 1 8
14 3 364
11 10 11
20 9 167960
22 3 1540
16 3 560
24 7 346104
13 10 286
15 8 6435
7 5 21
3 3 1
6 4 15
16 15 16
5 4 5
19 19 1
27 16 13037895
26 1 26
25 14 4457400
6 5 6
13 6 1716
17 7 19448
22 4 7315
27 6 296010
24 18 134596
19 9 92378
3 1 3
19 15 3876
9 5 126
15 10 3003
2 2 1
13 2 78
27 3 2925
17 2 136
15 8 6435
18 14 3060
24 21 2024
13 10 286
25 8 1081575
28 14 40116600
8 1 8
15 15 1
7 2 21
14 3 364
14 4 1001
2 2 1
4 4 1
23 1 23
29 2 406
15 7 6435
12 7 792

28 25 3276
23 20 1771
15 6 5005
8 1 8
9 2 36
28 11 21474180
13 3 286
4 2 6
27 12 17383860
25 16 2042975
8 3 56
21 20 21
29 27 406
21 18 1330
6 1 6
20 18 190
16 3 560
10 7 120
20 4 4845
6 1 6
4 3 4
30 15 155117520
14 6 3003
12 3 220
8 1 8
12 3 220
4 3 4
26 12 9657700
6 1 6
26 6 230230
15 1 15
29 10 20030010
22 16 74613
17 5 6188
4 4 1
19 11 75582
30 1 30
22 22 1
19 5 11628
31 2 465
2 2 1
21 21 1
6 5 6
10 6 210
2 2 1
11 5 462
14 12 91
22 10 646646
1 1 1
10 7 120
1 1 1
3 2 3
20 10 184756
24 21 2024
3 2 3

4 2 6
15 6 5005
24 14 1961256
30 8 5852925
26 13 10400600
8 2 28
11 2 55
12 3 220
17 12 6188
15 7 6435
16 9 11440
7 4 35
30 20 30045015
28 5 98280
26 21 65780
18 13 8568
18 5 8568
11 11 1
32 31 32
13 1 13
11 10 11
7 2 21
13 7 1716
29 22 1560780
22 9 497420
21 19 210
31 13 206253075
32 7 3365856
6 3 20
15 12 455
5 5 1
7 5 21
17 16 17
4 1 4
25 11 4457400
14 4 1001
8 6 28
31 4 31465
25 19 177100
7 6 7
14 12 91
29 4 23751
17 13 2380
29 4 23751
12 6 924
19 8 75582
6 6 1
21 21 1
14 3 364
14 13 14
21 5 20349
3 1 3
20 1 20
12 11 12
33 14 818809200

19 14 11628
17 12 6188
17 16 17
14 14 1
17 10 19448
31 6 736281
27 5 80730
8 8 1
16 8 12870
18 15 816
28 11 21474180
15 9 5005
19 2 171
18 10 43758
32 24 10518300
27 24 2925
15 14 15
28 1 28
35 18 4537567650
1 1 1
7 7 1
29 18 34597290
17 9 24310
14 8 3003
18 12 18564
3 1 3
10 8 45
1 1 1
19 13 27132
11 10 11
19 12 50388
19 10 92378
12 9 220
11 7 330
27 12 17383860
8 6 28
9 4 126
15 8 6435
18 16 153
9 3 84
31 21 44352165
5 3 10
27 20 888030
21 13 203490
6 2 15
5 4 5
26 18 1562275
24 14 1961256
19 3 969
26 20 230230
23 16 245157
8 4 70
30 18 86493225
35 15 3247943160
11 11 1

24 7 346104
6 4 15
22 16 74613
9 5 126
1 1 1
33 11 193536720
24 4 10626
6 5 6
13 3 286
25 2 300
34 13 927983760
24 24 1
1 1 1
16 5 4368
27 11 13037895
37 15 9364199760
28 1 28
20 5 15504
20 7 77520
19 16 969
33 28 237336
16 8 12870
2 2 1
37 31 2324784
4 3 4
13 2 78
8 6 28
18 18 1
17 12 6188
26 4 14950
35 33 595
4 2 6
33 27 1107568
13 11 78
31 30 31
39 34 575757
13 10 286
17 11 12376
35 4 52360
36 14 3796297200
21 7 116280
3 2 3
10 7 120
38 27 1203322288
26 20 230230
24 24 1
22 4 7315
22 15 170544
4 2 6
19 13 27132
1 1 1
7 6 7
11 3 165
11 1 11
29 13 67863915

39 11 1676056044
3 3 1
27 4 17550
20 20 1
31 9 20160075
26 19 657800
31 12 141120525
36 10 254186856
25 24 25
21 3 1330
40 34 3838380
35 29 1623160
13 12 13
35 29 1623160
28 26 378
12 7 792
19 19 1
17 10 19448
13 11 78
29 9 10015005
39 34 575757
19 13 27132
22 8 319770
1 1 1
20 20 1
14 4 1001
39 19 68923264410
2 2 1
18 18 1
14 10 1001
14 2 91
34 30 46376
16 10 8008
40 7 18643560
25 25 1
30 10 30045015
14 8 3003
10 9 10
40 24 62852101650
1 1 1
11 11 1
33 16 1166803110
1 1 1
7 2 21
19 9 92378
7 7 1
42 22 513791607420
19 6 27132
26 6 230230
31 17 265182525
36 15 5567902560
28 20 3108105
36 21 5567902560
21 15 54264
33 21 354817320

8 8 1
11 8 165
3 2 3
35 5 324632
1 1 1
38 34 73815
16 16 1
36 9 94143280
42 1 42
35 25 183579396
23 10 1144066
22 8 319770
3 2 3
10 8 45
23 23 1
10 8 45
31 30 31
14 13 14
25 6 177100
10 2 45
31 16 300540195
1 1 1
9 9 1
33 14 818809200
5 1 5
5 4 5
7 7 1
10 8 45
1 1 1
16 12 1820
1 1 1
24 23 24
13 9 715
37 15 9364199760
16 2 120
18 2 153
36 10 254186856
42 4 111930
22 16 74613
6 3 20
26 8 1562275
3 1 3
10 1 10
8 6 28
43 9 563921995
8 6 28
1 1 1
24 2 276
44 34 2481256778
32 24 10518300
31 25 736281
44 38 7059052
39 32 15380937
7 4 35
6 1 6

22 12 646646
42 2 861
8 5 56
20 8 125970
15 4 1365
10 10 1
10 6 210
39 2 741
30 23 2035800
22 2 231
19 12 50388
43 6 6096454
30 11 54627300
29 18 34597290
38 27 1203322288
31 7 2629575
3 2 3
20 4 4845
41 7 22481940
21 17 5985
35 32 6545
31 12 141120525
1 1 1
28 3 3276
3 2 3
18 7 31824
19 8 75582
9 6 84
39 37 741
26 26 1
12 11 12
36 4 58905
47 17 2741188875414
45 36 886163135
33 21 354817320
36 7 8347680
26 4 14950
38 30 48903492
9 9 1
31 2 465
37 13 3562467300
13 3 286
36 3 7140
2 2 1
38 13 5414950296
5 3 10
25 16 2042975
40 39 40
34 26 18156204
19 2 171
7 7 1
24 17 346104
22 8 319770
9 6 84
45 22 4116715363800

3 3 1
38 27 1203322288
25 2 300
7 6 7
15 10 3003
10 9 10
41 29 7898654920
33 6 1107568
35 35 1
32 32 1
19 15 3876
10 7 120
21 4 5985
39 24 25140840660
41 25 103077446706
14 6 3003
16 5 4368
15 9 5005
39 6 3262623
34 26 18156204
7 1 7
27 6 296010
23 10 1144066
37 29 38608020
11 5 462
16 8 12870
40 37 9880
26 11 7726160
17 17 1
24 1 24
35 15 3247943160
47 42 1533939
13 10 286
47 21 12551759587422
19 11 75582
42 37 850668
24 3 2024
36 9 94143280
3 3 1
37 2 666
12 7 792
1 1 1
20 4 4845
20 9 167960
2 1 2
41 35 4496388
10 3 120
11 5 462
9 4 126
15 15 1
24 18 134596
3 1 3
35 23 834451800
48 39 1677106640
46 19 4154246671960

44 12 21090682613
37 7 10295472
8 7 8
34 32 561
18 1 18
27 24 2925
2 1 2
48 46 1128
40 33 18643560
22 5 26334
26 3 2600
21 11 352716
49 41 450978066
51 19 48459472266975
6 1 6
28 24 20475
40 17 88732378800,0001
6 3 20
52 22 270533919634160
48 35 192928249296
18 17 18
28 13 37442160
41 4 101270
30 2 435
12 3 220
4 2 6
43 7 32224114
21 19 210
50 16 4923689695575
4 1 4
29 18 34597290
29 18 34597290
37 16 12875774670
29 11 34597290
30 10 30045015
29 28 29
2 2 1
47 23 16123801841550
33 10 92561040
22 16 74613
44 23 2012616400080
5 1 5
9 1 9
4 3 4
37 34 7770
6 5 6
48 18 7309837001104
9 1 9
1 1 1
51 7 115775100
7 4 35
38 30 48903492
5 2 10
44 21 2012616400080
53 9 4431613550

17 14 680
10 6 210
32 6 906192
34 5 278256
46 32 239877544005
53 2 1378
21 10 352716
21 18 1330
6 2 15
26 6 230230
28 13 37442160
10 10 1
8 4 70
28 21 1184040
48 13 192928249296
41 37 101270
1 1 1
47 28 6973199770790
39 33 3262623
46 35 13340783196
2 2 1
21 17 5985
44 26 1029530696964
23 22 23
50 39 37353738800
34 20 1391975640
38 1 38
14 5 2002
12 2 66
30 23 2035800
15 8 6435
37 9 124403620
21 10 352716
16 10 8008
47 38 1362649145
30 10 30045015
5 3 10
51 44 115775100
27 17 8436285
32 22 64512240
40 23 88732378800,0001
22 13 497420
27 16 13037895
23 1 23
50 48 1225
6 4 15
19 16 969
31 14 265182525
46 8 260932815
50 25 126410606437752
41 27 35240152720
50 49 50
24 18 134596
21 20 21
24 5 42504

11 11 1
4 2 6
2 2 1
7 7 1
34 12 548354040
38 8 48903492
8 8 1
12 7 792
4 4 1
20 6 38760
1 1 1
45 19 2438362177020
9 5 126
24 16 735471
18 7 31824
45 35 3190187286
54 18 96926348578605
23 14 817190
6 5 6
56 33 3167295784216200
21 18 1330
44 14 114955808528
22 19 1540
48 10 6540715896
21 9 293930
2 2 1
41 37 101270
37 6 2324784
46 15 511738760544
24 18 134596
24 1 24
38 37 38
46 40 9366819
44 34 2481256778
14 7 3432
22 9 497420
13 5 1287
29 20 10015005
16 3 560
26 25 26
15 6 5005
37 8 38608020
25 21 12650
42 12 11058116888
33 4 40920
38 22 22239974430
55 52 26235
53 35 64617565719070
55 21 841728816603675
44 29 229911617056
47 23 16123801841550
16 9 11440
51 35 7174519270695
28 15 37442160
18 4 3060

12 4 495
30 6 593775
39 28 1676056044
56 4 367290
56 39 97997533741800
26 22 14950
30 25 142506
26 10 5311735
11 1 11
25 11 4457400