Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
CSF13 – Fundamentos de Programação 1
Prof. Bogdan Tomoyuki Nassu
Profa. Leyza Baldo Dorini

1.

a)

```
/* A resposta deste exercício é bem simples, se você entendeu como se comportam
   loops aninhados. */

#include <stdio.h>

int main ()
{
    int i, j; /* Um contador para cada dado. */

    for (i = 1; i <= 6; i++)
        for (j = 1; j <= 6; j++)
            printf ("%d %d\n", i, j);

    return (0);
}
```

b)

```
/* A mudança é simples: basta colocar um loop a mais. A maior dificuldade aqui é
   trabalhar com os 3 loops aninhados. Este tipo de estrutura é comum a muitos
   problemas que envolvem descobrir permutações. */

#include <stdio.h>

int main ()
{
    int i, j, k; /* Um contador para cada dado. */

    for (i = 1; i <= 6; i++)
        for (j = 1; j <= 6; j++)
            for (k = 1; k <= 6; k++)
                printf ("%d, %d e %d\n", i, j, k);

    return (0);
}
```

c)

```
/* A modificação é MUITO SIMPLES. Basta trocar o início dos 2 loops internos: */

    for (i = 1; i <= 6; i++)
        for (j = i; j <= 6; j++)
            for (k = j; k <= 6; k++)
```

2.

```
/* Neste exercício, reaproveitamos uma parte da estrutura de um programa que já
   tínhamos visto: o que determinava se um dado número n era primo. Assim como
   naquela versão, o "miolo" da solução não é o mais otimizado, mas deixamos
   assim por simplicidade. */

#include <stdio.h>

#define N 30

int main ()
{
    int n, div, eh_primo;
    int total, acumulados;

    // Vamos verificando se cada n >= 2 é primo.
    // Repete até ter acumulado N valores.
    // Note como estamos usando um for para um loop indeterminado!
    acumulados = 0;
    total = 0;
    for (n = 2; acumulados < N; n++)
    {
        // Este "miolo" já apareceu anteriormente!
        eh_primo = 1;
        for (div = 2; div < n && eh_primo; div++)
            if (n%div == 0)
                eh_primo = 0;

        /* Uma versão melhor do "miolo" seria:
        eh_primo = 1;
        if (n > 2 && n % 2 == 0)
            eh_primo = 0;

        final = sqrt (n); // Teria que declarar um int final lá em cima também!
        for (div = 3; div <= final && eh_primo; div+=2)
            if (n%div == 0)
                eh_primo = 0; */


        if (eh_primo)
        {
            acumulados++;
            total += n;
        }
    }

    printf ("%d\n", total);

    return (0);
}
```

3.

```
#include <stdio.h>

int main ()
{
    int n, x, div, achou;

    scanf ("%d", &n);

    /* Força bruta! Testa valores de x até encontrar um que seja divisível por
       todos os números de 1 a n. Começamos de n porque já sabemos de antemão
       que o número é múltiplo de n. */
    achou = 0;
    x = n;
    while (!achou)
    {
        // Verifica se x é divisível por TODOS os números entre 2 e n.
        // Por 1 todo mundo já é divisível!!!
        for (div = 2; div <= n; div++)
            if (x % div != 0)
                break;

        // Se tiver passado por TODOS os divisores, o loop fez todas as
        // iterações, e div > n.
        if (div > n)
            achou = 1;
        else // Não achou, tenta outro x.
            x++;
    }

    printf ("%d eh divisivel por todos os numeros entre 1 e %d.\n", x, n);

    return (0);
}
```

4.

a)

```
/* A estrutura deste programa já tinha aparecido antes! */

#include <stdio.h>

int main ()
{
    int i, j, n;

    scanf ("%d", &n);

    // Para cada linha...
    for (i = 0; i < n; i++)
    {
        // ... mostra n-i As.
        for (j = 0; j < n-i; j++)
            printf ("%c", 'A' + j);
        printf ("\n");
    }

    return (0);
}
```

b) Basta modificar o printf do loop mais interno para:

```
printf ("%c", 'A' + i + j);
```

c) Mais uma mudança simples no printf do loop mais interno. O ponto essencial aqui é observar se você consegue mudar o padrão mostrado apenas com essas pequenas manipulações, sem criar montes de novas variáveis e estruturas! Para esta versão, o printf fica:

```
printf ("%c", 'A' + (n-i-1));
```

5.

```
/* A chave para entender este problema é enxergar o padrão. Desenhe os losangos
 para diferentes valores de n. Observe que a cada linha temos um número de
 espaços seguido de um número de caracteres visíveis. Na primeira linha, temos
 exatamente n espaços seguidos de 1 caractere visível. Para cada linha seguinte,
 o número de espaços à esquerda diminui em 1, e o número de caracteres visíveis
 aumenta em 2. Isso se repete até que tenhamos 0 espaços e 2n+1 caracteres
 visíveis. O número de espaços então volta a aumentar, até que no fim temos
 novamente n espaços à esquerda e 1 caractere visível. Este é o tipo de padrão
 que a imensa maioria NÃO VAI enxergar se simplesmente tentar fazer o código,
 sem planejar antes. O maior desafio aqui não é a sintaxe da linguagem C, e não
 tem relação com funções de bibliotecas ou com o conhecimento da linguagem. O
 grande desafio é enxergar o padrão que gera o losango.

 Existem várias soluções possíveis para este problema, mas todas elas envolvem
 controlar o número de espaços e de caracteres visíveis que precisam ser
 impressos a cada linha. Eu poderia, por exemplo, ter inicializado n_espacos e
 direcao_espacos dentro do for, e ter feito o incremento ali também. Poderia
 também ter controlado separadamente o número de caracteres visíveis. Poderia
 ter usado dois loops separados, um enquanto o número de espaços diminui e outro
 enquanto aumenta. E por aí vai. É bom também imaginar o quanto o código ficaria
 confuso se as variáveis tivessem nomes como a ou b. */

#include <stdio.h>

int main () {
    char caractere; /* O caractere especificado pelo usuário. */
    int n; /* O parâmetro usado para calcular a largura máxima. */
    int largura_max; /* A largura máxima do losango (no centro). */
    int n_linha; /* Número da linha atual. */
    int n_espacos; /* Número de espaços que queremos em uma linha. */
    int direcao_espacos; /* Sempre 1 ou -1, nos diz se o número de espaços vai
                            aumentar ou diminuir a cada iteração. */

    scanf ("%c %d", &caractere, &n);

    largura_max = n*2+1; /* Largura máxima e altura do losango. */
    n_espacos = n; /* Começamos com n espaços. */
    direcao_espacos = -1; /* Começamos diminuindo o número de espaços em 1 a
                             cada iteração. */
    for (n_linha = 0; n_linha <= largura_max; n_linha++) {
        int i; /* Um contador genérico. */

        /* Imprime um punhado de espaços. */
        for (i = 0; i < n_espacos; i++)
            printf (" ");

        /* Imprime caracteres visíveis. O número de caracteres é dado pela
           largura total menos o dobro do número de espaços. */
        for (i = 0; i < largura_max - (n_espacos*2); i++)
            printf ("%c", caractere);

        printf ("\n"); /* Terminamos esta linha! */

        if (!n_espacos) /* Se o número de espaços chegou a 0, vamos começar a
                           aumentar o número de espaços. */
            direcao_espacos = 1;

        n_espacos += direcao_espacos; /* Atualiza o número de espaços. */
    }

    return (0);
}
```