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

(*) 1.

a) Escreva um programa que mostre todos os resultados possíveis quando arremessamos 2 dados de 6 faces. A ordem aqui é relevante, portanto 1, 2 é um resultado diferente de 2, 1, e ambos devem ser mostrados. Dica 1: se achar muito difícil, escreva em algum lugar todas as combinações e veja se existe algum padrão. Dica 2: se ainda assim achar muito difícil, comece mostrando somente todos os resultados possíveis quando arremessamos um único dado!

b) Modifique o programa para que ele mostre todos os resultados possíveis quando arremessamos 3 dados de 6 faces. A ordem aqui é relevante, portanto 1, 1, 2 é um resultado diferente de 2, 1, 1, e ambos devem ser mostrados.

c) Modifique novamente o programa para que ele só mostre cada combinação de números uma única vez. Ou seja, se ele apresentar 1, 1, 2, não deve mostrar 2, 1, 1, nem 1, 2, 1. Dica: a modificação é **muito simples**. Se você não conseguir enxergar a solução facilmente, faça de conta que os dados têm apenas 3 faces e escreva em um papel todas as possibilidades possíveis para a versão a) - serão 27 possibilidades - e risque as possibilidades redundantes - devem restar apenas 10. Verifique então se as possibilidades restantes formam algum padrão.

(***) 2. Um número positivo é primo se for divisível somente por 1 e por ele mesmo. Escreva um programa que calcula a soma dos N primeiros números primos, onde N é uma constante declarada em uma macro. Por exemplo, como os 5 primeiros números primos são: 2, 3, 5, 7 e 11, para N = 5 a soma será 28.

(***) 3. Escreva um programa que recebe como entrada um número positivo *n* e mostra qual é o menor número que é divisível por todos os números de 1 a *n*. Por exemplo, para *n* = 10, o programa deve mostrar o número 2520, que é divisível por todos os números entre 1 e 10.

(***) 4.

a) Faça um programa que, dado um valor *n* informado pelo usuário, imprima uma pirâmide de caracteres com *n* linhas seguindo o padrão abaixo. Considere que 1 ≤ n ≤ 26 (o programa não precisa testar isso!).

Exemplo para *n* = 3:

```
A B C
A B
A
```

Exemplo para *n* = 5:

```
A B C D E
A B C D
A B C
A B
A
```

b) Modifique o seu programa para imprimir o padrão como exemplificado abaixo. Dica: se a sua solução para a questão anterior for eficiente, a mudança é muito simples!

Exemplo para *n* = 3:

```
A B C
B C
C
```

Exemplo para *n* = 5:

```
A B C D E
B C D E
C D E
D E
E
```

c) Modifique o seu programa para imprimir o padrão como nos exemplos abaixo. Dica: se a sua solução para a questão anterior for eficiente, a mudança é muito simples!

Exemplo para *n* = 3:

```
C C C
B B
A
```

Exemplo para *n* = 5:

```
E E E E E
D D D D
C C C
B B
A
```

(☠) 5. Escreva um programa que imprima losangos formados por caracteres ASCII. O usuário deve especificar o caractere que será usado, e fornecer um número *n*. A largura do losango no centro deve ser igual a *2n+1* caracteres. A cada linha acima ou abaixo do centro, a largura é reduzida em 2 caracteres. Para manter o formato do losango, use espaços antes dos caracteres visíveis. Por exemplo, se o caractere é '#' e *n* é igual a 2, a saída do programa é:

```
  #
 ###
#####
 ###
  #
```

Se o caractere é '@' e *n* é igual a 5, a saída do programa é:

```
      @
     @@@
    @@@@@
   @@@@@@@
  @@@@@@@@@
@@@@@@@@@@@
 @@@@@@@@@
  @@@@@@@
   @@@@@
    @@@
     @
```

Dica: A chave para descobrir como resolver este exercício é se desligar da linguagem C, e procurar um padrão em como os losangos são gerados. A solução para este problema precisa começar no papel. A ideia por trás do algoritmo é bem mais sofisticada que o código que a implementa.