Lista geral de exercícios
Prof. João B. Oliveira

Esta lista de exercícios serve para disciplinas diferentes e deve ser usada de maneiras diferentes:

- Em disciplinas de Algoritmos geralmente não é necessário implementar o que foi desenvolvido, por isso vários exercícios podem ser ignorados (mas fazê-los é uma ótima maneira de aprender como seus algoritmos funcionam quando vão para um computador).
- Em disciplinas de Programação, espera-se que os exercícios sejam realmente programados e testados!

1 Exercícios sequenciais

1.0 Escreva um programa em C que não faz nada:

```
void main() {
}
```

Tudo que está aí em cima é obrigatório, e as ações que devem ser feitas devem ser colocadas entre as chaves { e }. Como não há nada entre as chaves, o programa está correto mas não faz nada.

1.1 Agora que você tem um programa, adapte-o para imprimir uma mensagem:

```
#include <stdio.h>
void main() {
  printf(“Bom diaaaaa!”);   // Aspas duplas e ;
}
```

Desejamos imprimir uma mensagem na tela usando a função printf(). Para isso precisamos também da linha que inicia com #include, que significa “pegue as funções da biblioteca stdio para usar neste programa”, pois a função printf() está na biblioteca stdio e o programa precisa ser informado disso. O include tem que aparecer antes que as funções sejam usadas no programa.

1.2 Escreva um programa que cria alguns inteiros e floats e depois imprima os valores que você colocou neles. Por exemplo, teste o trecho de programa abaixo. Comece com um único comando `printf` e vá colocando cada um dos outros e rodando para ver a diferença:

```
void main() {
  int a = 105;
  float x = 101.31;
  printf(“Com d: %d\n”, a );     // Depois de rodar cada um dos comandos,
  printf(“Com 7d: %7d\n”, a );   // explique o que ele fez!
  printf(“Com 07d: %07d\n”, a );
  printf(“Com -7d: %-7d\n”, a );
  printf(“Com f: %f\n”, a );
  printf(“Com 7f: %7f\n”, x );
  printf(“Com .3f: %.3f\n”, x );
  printf(“Com 7.4f: %7.4f\n”, x );
  printf(“Com c: %c\n”, a );
  printf(“Com c: %c\n”, x );
}
```

A primeira vez que você testar este programa ele deve dar um problema. Descubra o que aconteceu e resolva!

1.3 Escreva um programa em C que apenas imprime duas variáveis inteiras a e b, sem dar valores a elas. Rode o programa mais de uma vez e veja o que acontece com os valores que aparecem. Eles são sempre zero? Eles são sempre iguais? Eles mudam quando você roda o programa várias vezes?

1.4 Escreva um programa que declara duas variáveis inteiras a e b, dando valores a elas e imprimindo os valores. Depois adicione comandos para que o programa troque os valores das variáveis passando o valor de a para b e vice-versa. Imprima de novo para confirmar que deu tudo certo. Seu programa deve ser parecido com este aqui:

```
int a = 5;
int b = 11;
printf("a: %d,   b: %d\n", a, b );   // Imprime 5 e 11
...
// Comandos misteriosos que trocam a com b
...
printf("a: %d,   b: %d\n", a, b );   // Imprime 11 e 5   (não é para imprimir b
```

1.5 Escreva um programa que faz conversão de temperaturas de Fahrenheit para Celsius, segundo a fórmula conhecida:

$$C = 5/9 * (F - 32)$$

Deu tudo certo? Se não der certo, deve ter uma explicação. Ou você pode tentar alguma das fórmulas alternativas abaixo:

$$\begin{aligned} C &= (F - 32) * 5/9 \\ &= 5/9.0 * (F - 32) \\ &= 1.0 * 5/9 * (F - 32) \\ &= 5 * 1/9 * (F - 32) \end{aligned}$$

1.6 Escreva um programa que recebe as dimensões (em metros) de um terreno retangular e em seguida as dimensões de uma casa (também em metros e retangular) sobre este terreno. Em seguida calcule e apresente a área livre do terreno, em metros quadrados e em percentagem de terreno livre.

1.7 Altere seu programa do exercício 1.6 para funcionar com uma casa redonda em um terreno redondo. A linguagem C não "conhece" π, então você pode criar uma variável float e colocar 3.141592 nela.

1.8 Descubra o que faz o programa abaixo. Primeiro simule em uma folha de papel o que vai acontecer e depois rode o programa para confirmar:

```
int a = 5;
int b = 9;
printf("a = %d, b = %d\n", a, b );
a = a + b; // Quanto valem a e b agora?
b = a - b; // E agora?
a = a - b; // E agora?
printf("a = %d, b = %d\n", a, b );
```

1.9 Escreva um programa para um terminal de banco, que recebe um valor inteiro R e determina o número de notas de 100, 50, 10, 5 e 1 reais necessárias para pagar a quantia R. Faça de forma que o número de notas usado seja o menor possível.

1.10 Altere seu programa anterior para que ele trabalhe com notas de 110, 50 e 10 reais e descubra que ele não funciona bem para dar troco de 150 reais, por que deve dar um troco com 5 notas e poderiam ser apenas 3.

1.11 Escreva um programa que recebe três números inteiros e depois imprime quantos desses números eram ímpares. Depois disso adapte seu programa para dizer quantos números são pares. Pode ser útil usar a operação %, que é o resto de divisão.

1.12 Escreva um programa que recebe um horário (horas, minutos, segundos) e determina quantos segundos já se passaram desde que o dia começou.

1.13 Altere seu programa anterior para também dizer quanto tempo falta até o fim do dia. Dê o tempo que falta em segundos, e depois apresente esse tempo em horas, minutos e segundos.

1.14 Escreva um programa que recebe o horário atual (horas, minutos, segundos) e mais um número de segundos, e diz que horas serão depois que estes segundos tiverem passado.

1.15 Se você estiver a uma altura h (em metros) sobre o nível do mar, sua distância d (também em metros) até o horizonte é dada bem aproximadamente pela fórmula

$$d = \sqrt{h^2 + 2rh},$$

onde r é o raio da Terra (cerca de 6378150 metros)[1]. Escreva um programa que recebe uma altura h e determina a que distância está o horizonte, sabendo que em C a função que calcula raizes quadradas se chama `sqrt()` e para usá-la você tem que usar a biblioteca `math.h`. Para matar a curiosidade, aqui estão as alturas de alguns locais interessantes:

Local	Altura
Torre de Pisa (topo)	55 m
Taj Mahal (topo da cúpula)	65 m
Foguete Saturno 5	110 m
Torre Eiffel (plataforma 3)	276 m
Pão de Açúcar	394 m
World Trade Center (torre 1)	417 m
Cristo Redentor	704 m
Mont Blanc	4810 m
Monte K2	8611 m
Monte Everest	8830 m
Avião	20 m a 12.000 m

1.16 Escreva um programa que recebe dois pontos no plano, $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ e calcula:

(a) A distância entre eles, dada por $d(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

(b) A inclinação da reta $y = ax + b$ que une estes dois pontos, dada por $a = (y_2 - y_1)/(x_2 - x_1)$. Existe uma situação especial em que seu programa pode apresentar um erro. Identifique o erro.

[1] Atenção: esta fórmula é uma aproximação apenas, baseada em geometria simples e sem levar em conta uma série de fatores atmosféricos. Também perceba que se você usar as unidades em metros os números serão bastante grandes, podendo causar problemas.

1.17 Você resolveu dar uma força para a NASA, que deseja calcular o peso das pessoas em diferentes planetas do sistema solar. (Isso vai ter importantes consequencias na colonização dos planetas pela humanidade: os gordinhos vão preferir planetas onde baixem de peso, os magrelinhos vão querer o contrário). Escreva um programa que recebe o peso de uma pessoa e calcula o novo peso em vários corpos celestes, de acordo com as constantes gravitacionais de cada planeta (estas constantes estão normalizadas em relação à constante da Terra), dadas abaixo:

Corpo celeste	Fator de conversão
Mercúrio	0.3649337410
Vênus	0.9041794087
Marte	0.3812436289
Lua	0.1651376146
Terra	1
Júpiter	2.6513761467
Saturno	1.1386340468
Urano	1.0693170234
Netuno	1.3506625891
Plutão	0.2252803261

Depois de fazer as conversões, imprima o quanto as pessoas pesariam nos seus novos planetas.

2 Exercícios com decisão

2.0 Escreva um programa que recebe dois números a e b e depois imprime o menor dos dois números.

2.1 Escreva um programa que recebe três números a, b e c e depois imprime o menor dos três números.

2.2 Escreva um programa que recebe três números a, b e c e depois verifica se os três foram dados em ordem crescente.

2.3 Uma loja de roupas chiques tem uma promoção especial: você leva dois produtos e tem um desconto de 50% no produto mais barato. Escreva um programa que recebe o preço de dois produtos e calcula o valor final a ser pago. Tome cuidado por que os valores dos produtos podem ser dados em qualquer ordem.

2.4 A loja do exercício 2.3 acabou de melhorar a promoção! Agora, ao levar três produtos você ganha 50% de desconto no mais barato e 25% de desconto no segundo mais barato! Escreva um programa que recebe o preço de três produtos e calcula o valor final a ser pago. Claro, não esqueça de que os produtos podem passar no caixa em qualquer ordem!

2.5 O ingresso do parquinho é de 5 reais para crianças de 7 anos ou menos, 10 reais para pessoas entre 8 e 18 anos e 12 reais para maiores de 18, mas custa apenas 8 reais para idosos (acima de 59). Escreva um programa que recebe a idade do visitante e informa o valor a ser pago.

2.6 Escreva um programa que recebe três números a, b e c que podem estar em qualquer ordem e depois imprime o valor que estaria no meio se eles fossem escritos em ordem crescente.

2.7 Escreva um programa que recebe três números a, b e c que representam lados de um triângulo e dá a mensagem adequada:

(a) "É um triângulo equilátero";

(b) "É um triângulo isósceles";

(c) "É um triângulo escaleno";

(d) "Não é um triângulo";

2.8 Adapte seu programa do exercício 1.16 para receber as coordenadas dos centros de dois círculos, junto com seus raios. Em seguida seu programa deve determinar se os dois círculos se intersectam ou não. Depois de conseguir isso, adapte seu programa para não precisar usar uma raiz quadrada.

2.9 Escreva um programa que recebe três números a, b e c e informa se pode existir um triângulo retângulo com estes lados.

2.10 Você resolveu renegociar seu financiamento estudantil e as regras do banco são estas:

(a) Você precisa dar uma entrada para renegociar. Ela deve ser de 10% da dívida ou mil reais (o que for menor).

(b) O saldo pode ser pago em até 72 vezes, mas as parcelas não podem ser de menos de 200 reais (só a última, para completar o valor).

Com estas informações, escreva um programa que recebe o valor da sua dívida e o número de parcelas em que deseja pagar e responde se é possível aceitar seu plano de pagamento ou não. Se for possível aceitar, seu programa deve informar o valor da entrada e o valor de cada prestação a ser paga.

Depois refaça o exercício em uma nova versão: o programa recebe a dívida e deve informar o maior número de prestações possível para o pagamento da dívida.

2.11 Escreva um programa que recebe os coeficientes a, b e c de um polinômio de segundo grau $ax^2 + bx + c$ e determina para quais valores de x este polinômio é igual a zero. Tome cuidado para calcular todas as situações corretamente e avise o usuário se houverem situações de erro.

2.12 Escreva um programa que é capaz de escrever os dígitos de um número de até três dígitos. Por exemplo, ao receber 358 seu programa deve escrever "**três cinco oito**".

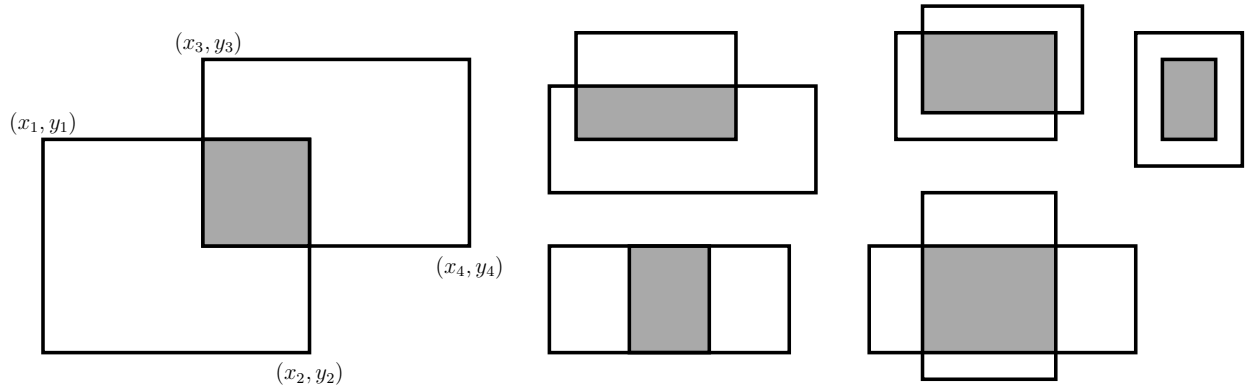
2.13 Faça um programa que recebe um ano e verifica se o ano é bissexto, imprimindo uma mensagem. As regras para um ano ser bissexto são:

(a) Ele deve ser divisível por quatro.

(b) Se for um ano terminado em 00, então ele também deve ser divisível por 400.

2.14 Escreva um programa que recebe uma data (dia e mês) e diz quantos dias já se passaram desde o início do ano. Cuidado, pois os meses não tem sempre 30 dias!

2.15 Escreva um programa que recebe as coordenadas de dois retângulos (sempre dadas por seus cantos superior esquerdo e inferior direito como na figura abaixo) e diz se eles tem uma intersecção. Se ela existir, o programa também deve informar a sua área.

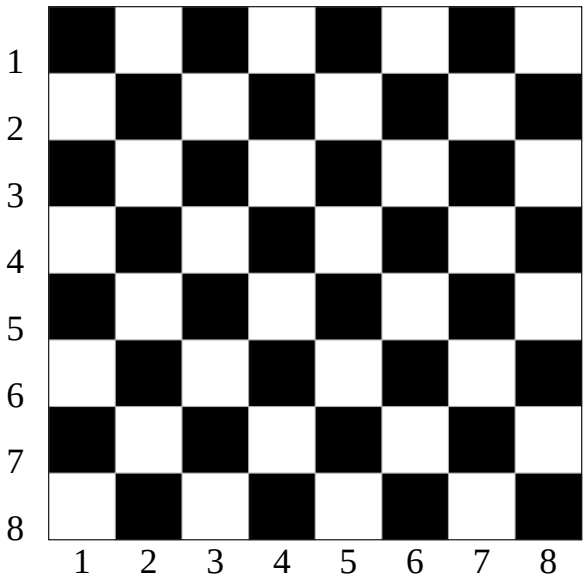


2.16 Escreva um programa que recebe duas datas do mesmo ano (ou seja, recebe apenas meses e dias) e calcula quantos dias existem de diferença entre as duas datas. Para facilitar, considere que o ano não é bissexto.

2.17 Refaça o programa do exercício 2.16, adaptando-o para funcionar com datas em anos diferentes. Você pode assumir que os anos sempre tem 365 dias.

2.18 Você pode abrir uma caderneta de poupança depositando o valor a no primeiro mês e o valor b nos meses seguintes. Supondo que a taxa de juros que você recebe pelo seu dinheiro é j (com $j > 1$, esperamos...) escreva um programa que lista quanto dinheiro você terá na conta a cada mês. Ache o valor correto de j, depois suponha que $a = b$ e faça as contas pra ver quanto você pode colocar na sua poupança todo mês. Depois descubra quando você terá um milhão.

2.19 Escreva um programa que recebe a linha e a coluna de uma casinha do tabuleiro de xadrez ao lado e diz qual a cor da casa.
Depois de fazer isso escreva outro programa para receber as coordenadas de **duas** casas e dizer se são da mesma cor.



2.20 Escreva um programa que recebe as quatro notas da avaliação de um aluno (três provas e uma nota dos trabalhos) e apresenta os resultados:

(a) A média do aluno (supondo que todas as notas tem pesos iguais).

(b) Se a última nota era zero, o programa assume que esta nota ainda não existe e diz ao aluno qual nota ele deve tirar para ter média sete.

2.21 Na competição de mergulho olímpica as notas de um atleta são dadas assim: oito juízes dão notas entre 0 e 10. Em seguida a nota mais baixa e a mais alta são descartadas e a nota final do atleta é dada pela média das seis notas restantes. Faça um programa que recebe as oito notas correspondentes a um atleta e apresenta a sua nota final.

2.22 Escreva um algoritmo que recebe um ano maior do que zero e diz a que século ele pertence. Para relembrar, os anos de 1 a 100 são do século I, os anos de 101 a 200 são do século II e assim por diante. Você não precisa escrever o número do século com numeração romana, mas precisa **conferir** seu programa pra ver se ele está mesmo certo.

2.23 Na ginástica olímpica as notas de uma equipe são dadas assim: seis atletas fazem seus exercícios e recebem suas notas entre 0 e 10. Em seguida a nota geral da equipe é dada pela média das quatro maiores notas recebidas. Faça um programa que recebe seis notas correspondentes aos atletas e apresenta a nota geral da equipe.

2.24 Depois de fazer o primeiro contato com uma civilização extraterrestre (os termopilitas) você ficou encarregado de fazer a conversão de datas entre as duas civilizações. Por coincidência o ano dos termopilitas tem a mesma duração do ano terrestre, mas é dividido de maneira diferente: a menor unidade de tempo é o *blip*, que dura 3 segundos terrestres. Quarenta *blips* fazem um *tak*, que é a próxima unidade. Quinze *taks* fazem um *salek*, e 30 *saleks* fazem um *mazel*. Oito *mazel* fazem um *brat*, e 73 *brat* formam o ano inteiro. Com estas informações, resolva:

- Escreva um programa que recebe uma data terrestre no formato *dia mes hora min seg* e imprime a data no formato termopilita: *brat mazel salek tak blip*.
- Faça um outro programa que faz a conversão contrária, do ano termopilita para o ano terrestre.
- Aproveite e construa uma tabela de conversões das unidades de tempo termopilitas, dizendo o quanto cada uma delas vale nas unidades terrestres.

2.25 Em outubro de 1793 a recém instituída República Francesa colocou em vigor um novo calendário, que foi usado até 1806. Neste calendário o ano era dividido em 12 meses de exatamente 30 dias cada um, seguidos de 5 dias "sem mês" (eram seis dias em anos bissextos). Os nomes destes 12 meses eram:

(1) Vendémiaire (vindima)	(5) Pluviôse (chuvoso)	(9) Prairial (pastagem)
(2) Brumaire (nevoeiro)	(6) Ventôse (ventoso)	(10) Messidor (colheita)
(3) Frimaire (geada)	(7) Germinal (semeadura)	(11) Thermidor (calor)
(4) Nivôse (nevado)	(8) Floréal (florescência)	(12) Fructidor (frutuoso)

Além disso, cada mês era dividido em três *décadas* de 10 dias cada, e o décimo dia era um dia de descanso (isso não deixou o calendário mais querido, pois no calendário antigo o dia de descanso acontecia a cada sete dias). Os dias eram chamados de Primeiro até Décimo e eram divididos em 10 horas de 100 minutos, cada uma dividida em 100 segundos (nenhum desses tinha a mesma duração das unidades que temos hoje). Os cinco/seis dias sem mês que completavam o ano eram chamados de Dia da Virtude, do Gênio, do Trabalho, da Razão, da Recompensa e da Revolução e eram sempre feriados.

Sabendo tudo isso, escreva um algoritmo que faz a conversão de datas entre os calendários.

3 Exercícios com repetição

3.0 Escreva um programa que recebe dois números inteiros a e b e acha a soma de todos os inteiros entre a e b. Faça seu programa funcionar até mesmo se a for maior do que b.

3.1 Repita o exercício anterior, mas ache apenas a soma dos números que **não são** múltiplos de sete.

3.2 Repita o exercício anterior, mas em vez de achar a soma, ache a média dos números.

3.3

3.4 Escreva um programa para calcular a soma da progressão aritmética que inicia em a, tem razão b e com n termos:

$$a + (a + b) + (a + 2b) + (a + 3b) + \ldots$$

3.5 Escreva um programa que recebe um inteiro n e imprime a soma dos dígitos de n.

3.6 Escreva um programa que recebe um número n e depois aplica estar regras:

(a) Se n for par, ele é transformado em $n/2$

(b) Se n for ímpar ele é transformado em $3n + 1$

Faça isso até n virar 1 e vá imprimindo os números por onde ele passa. Ao final informe quantas transformações foram feitas até ele virar 1.

3.7 Refaça o programa do exercício 2.18 usando repetições e veja como a sua vida vai ficar mais fácil.

3.8 Escreva um programa que examina um número dado pelo usuário e diz se há um "33" dentro dele.

3.9 Escreva um programa que examina todos os números entre 1 milhão e 10 milhões e conta quantos números não tem "33" dentro deles.

3.10 Escreva um programa que recebe um inteiro n e diz se os dígitos de n estão em ordem crescente ou não.

3.11 Escreva um programa que recebe um valor n e imprime na tela

```
1
2 2
3 3 3
4 4 4 4
....
n n n n n n
```

Depois de fazer isso, altere seu programa para achar a soma de todos os números que foram impressos.

E depois disso, mude seu programa para imprimir a soma de cada linha depois dela ser impressa.

3.12 Escreva um programa que recebe 10 inteiros fornecidos pelo usuário e depois informa a média dos valores pares e a média dos valores ímpares.

3.13 Escreva um programa que imprime uma tabuada até o 10×10.

3.14 Escreva um programa que recebe um valor inteiro n e diz se este valor é um número primo. Se não for, o programa apresenta uma lista dos números que dividem n.

3.15 Escreva um programa para calcular a soma abaixo com 10, 20 e 30 termos, sem usar a função `pow()`:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots$$

3.16 Refaça o programa do exercício 2.16, adaptando-o para funcionar com datas em anos diferentes e incluindo anos bissextos com a informação do exercício 2.13. Use-o para determinar há quantos dias você nasceu. Teste os programas dos seus colegas e confirme que todos dão os mesmos valores.

3.17 Escreva um algoritmo que recebe dois números inteiros de quatro dígitos (por exemplo, 1743 e 5218) e cria um novo número que foi feito intercalando os dígitos dos dois números. Neste caso, o resultado para 1743 e 5218 seria 15724138. Esteja preparado, por que fazer isso não é muito simples.

3.18 A soma

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$$

cresce devagar, mas tem valores cada vez maiores e vai para ∞ à medida que n cresce. Escreva um programa que recebe um valor de n e apresenta o valor da soma ate incluir $1/n$.

3.19 Em um Universo Paralelo, o Deus supremo decide:

(a) No dia 1, vou colocar um coelhinho no planeta Zorg, que está vazio.

(b) No dia 2, vou colocar mais um coelhinho no planeta Zorg.

(c) A partir do dia 3 vou colocar no planeta Zorg a quantidade de coelhinhos que havia 2 dias antes.

Escreva um programa que diz quantos coelhinhos estão em Zorg no dia n.

3.20 Escreva um programa que lê uma série de números, imprimindo a média dos números que já entraram. O programa inicia perguntando quantos números serão digitados mas **não pode** armazenar os valores que vão chegando. Por exemplo:

Número	Média
2	2
5	3.5
7	4.666
2	4

Depois adapte seu programa para não perguntar mais quantos números serão digitados, faça ele ler números até que seja dado um número negativo.

3.21 Escreva um programa que recebe um inteiro n e escreve n com separadores a cada 3 dígitos. Por exemplo, se $n = 123456789$ então seu programa deve escrever 123.456.789. Depois teste seu programa com os números 9, 19, 109, 1009 e 10009.

3.22 Escreva um programa que determina o maior divisor comum de dois números a e b, usando o algoritmo de Euclides, que na versão abaixo deixa o valor do maior divisor comum em a.

```
enquanto b for diferente de zero
    tmp = a;
    a = b;
    b = tmp % b;
print a
```

3.23 Escreva um algoritmo para receber um inteiro n e escrever o valor de n convertido para a base 2. Depois disso mude seu programa para também receber uma base b e escrever o valor de n convertido para a base b. Note que b deve ser maior do que 1 (embora existam também conversões para bases negativas).

3.24 Um método simples de multiplicar dois números inteiros a e b é usado no interior da Rússia até hoje. Adaptando-o para a Era da Tecnologia da Informação, o algoritmo pode ser o seguinte:

(a) Cria-se um inteiro x, inicializado com zero.

(b) Se a é ímpar, soma-se b em x.

(c) Multiplica-se b por 2.

(d) Divide-se a por 2 (arredonda-se para baixo se for necessário).

(e) Repete-se os passos anteriores (exceto o primeiro, obviamente) até que a seja zero.

Ao final deste algoritmo, x estará com o valor de $a * b$. Suas tarefas são:

- Programar o algoritmo e confimar que ele funciona.
- Explicar por que ele funciona.

3.25 Escreva um programa que acha as soluções inteiras da equação

$$3X + 2Y - 7Z = 5$$

para valores de X, Y e Z entre zero e 100. Prepare-se, pois são algumas dúzias de soluções.

3.26 A Lapônia vai estabelecer um sistema com apenas três tipos de moedinhas de centavos lapões e o rei ja decidiu que não vai haver uma moeda de um centavo. Os ministros dizem que com essa decisão alguns valores podem ser impossíveis de fornecer como troco, e você deve escrever um algoritmo que recebe três valores propostos para as moedinhas e descobre quais os valores entre 1 e 100 centavos que **não** podem ser construídos com elas. Forneça os resultados para os seguintes casos de teste:

- Moedas de 2, 3 e 5 centavos.
- Moedas de 3, 4 e 7 centavos.
- Moedas de 2, 4 e 7 centavos.
- Moedas de 3, 7 e 11 centavos.
- Moedas de 2, 4 e 31 centavos.
- Moedas de 5, 7 e 17 centavos.
- Moedas de 5, 13 e 17 centavos.
- Moedas de 6, 9 e 19 centavos.
- Moedas de 7, 9 e 19 centavos.

3.27 Um método para cálculo de raizes quadradas de um número N já era conhecido pelos babilônios em... bom, há muito tempo atrás. (Este método também é conhecido como método de Heron, um matemático grego que o descreveu 20 séculos depois dos babilônios, perto do ano 50 DC.) Começando de um valor inicial k (geralmente valendo um) os babilônios geravam um novo valor de k de acordo com a regra:

$$k = \frac{k + N/k}{2}.$$

recalculando k várias vezes. À medida que o processo é repetido, os novos valores de k se aproximam cada vez mais da raiz de N. Escreva um programa que recebe o valor de N e imprime os primeiros doze valores obtidos com esta fórmula, testando-os para ver se eles realmente se aproximam da raiz de N.

3.28 Um dos métodos que determina raizes **cúbicas** de N é o método de Joãozinho, bastante semelhante ao de Heron. Iniciando com $k = 1$, o método segue a regra

$$k = \frac{2k + N/k^2}{3}.$$

Escreva um programa que calcula raizes cúbicas com este algoritmo. Adapte seu programa para que ele confirme os resultados!

3.29 Com a regra que você pode perceber a partir das equações abaixo, escreva um programa que calcula o valor de S_n para qualquer valor de n maior do que zero:

$$\begin{aligned} S_1 &= 1 \\ S_2 &= 2 + 3 \\ S_3 &= 4 + 5 + 6 \\ S_4 &= 7 + 8 + 9 + 10 \\ S_5 &= 11 + 12 + 13 + 14 + 15 \end{aligned}$$

3.30 Escreva um programa que recebe um número $x < 1$ e apresenta uma soma de frações que aproxima x. Por exemplo, se recebermos $x = 0.543$ uma soma de frações que pode aproximar x seria

$$x \approx 1/2 + 1/24 + 1/750 = 0.5429999999999$$

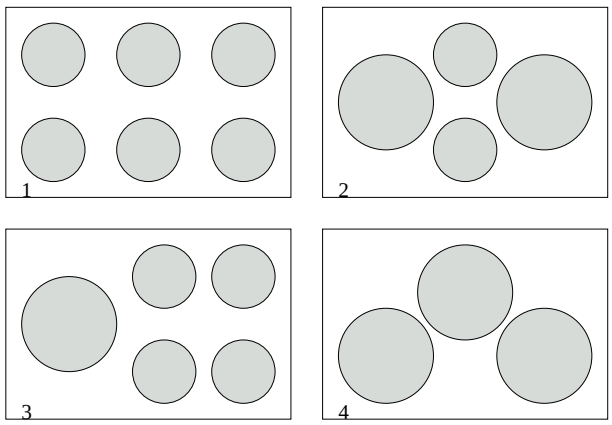
Seu programa deve ter algum tipo de controle para o grau de aproximação desejado pelo usuário.

3.31 Seus colegas da Biologia criaram uma bactéria que come lixo, consumindo cerca de metade de seu peso em lixo todos os dias. (Para facilitar, vamos supor que uma bactéria pesa cerca de 10^{-12} gramas). Seus colegas tem apenas cinco dessas bactérias, mas elas se dividem em duas todos os dias. A partir daí, escreva um programa que imprime uma tabela a partir do primeiro dia, quando existem apenas cinco bactérias, até dois meses depois (você terá de usar inteiros longos ou números de ponto flutuante para isso). A tabela deve listar o número do dia, quantas bactérias existem até o momento e quanto lixo elas comeram naquele dia. Depois de fazer isso, adapte seu programa para estas situações:

(a) Suponha que as bactérias vivem apenas cinco dias. Neste caso, as bactérias que morrem devem ser descontadas do total.

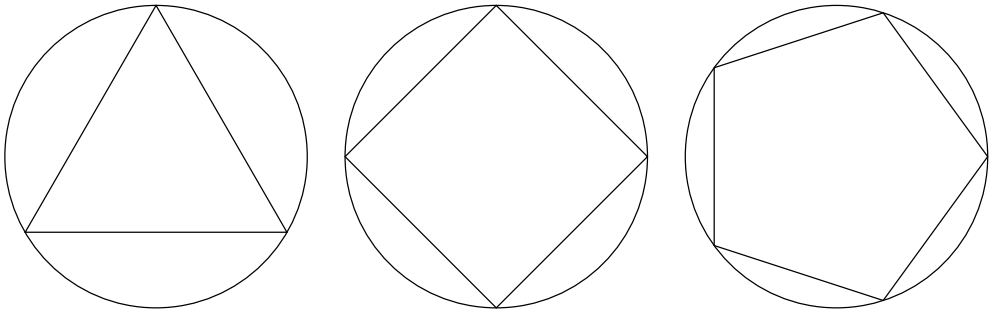
(b) Faça seu programa se adaptar às unidades de peso: de acordo com a quantidade de lixo consumida as unidades devem ser escritas em microgramas (10^{-6} g), miligramas (10^{-3} g), gramas, quilogramas ou toneladas, usando a maior unidade onde o peso seja maior do que um. Por exemplo, se for preciso imprimir 0.0037 gramas, você deve escolher 3.7 miligramas, e não 3700 microgramas.

3.32 Uma fábrica de panelas pode cortar de 4 maneiras diferentes as chapas de aço usadas para fazer panelas grandes e pequenas, e essas maneiras são ilustradas a seguir:



Cada corte permite a produção de um número diferente de panelas, e o custo de cortar a chapa 1 é de R$ 12, o custo de cortar a chapa 2 é de R$15, o custo de cortar a chapa 3 é de R$ 9 e o custo de cortar a chapa 4 é de R$ 18. Neste caso, escreva um algoritmo que encontra a maneira mais barata de produzir **pelo menos** 500 panelas pequenas e 200 panelas grandes. Ah, sim: você tem 50 chapas de cada tipo que pode (ou não) usar.

3.33 Você sempre pode desenhar um polígono regular dentro de um círculo se ele tiver três ou mais lados, como na figura abaixo:



Três lados Quatro lados Cinco lados

Supondo que os círculos tem raio um, escreva um programa que recebe um número $n \geq 3$ que representa o número de lados e faz as seguintes operações:

(a) Determina o comprimento de um lado do polígono.

(b) Determina o perímetro do polígono. Qual o valor do perímetro à medida que n cresce?

(c) Determine a área interna do polígono. Pode ser útil saber a área de um triângulo onde são conhecidos os lados a, b e c (por coincidência, esta fórmula foi descoberta pelo mesmo Heron do exercício 3.27...). Se $s = (a + b + c)/2$, então a área é dada por

$$A = \sqrt{s * (s - a) * (s - b) * (s - c)}.$$

(d) Determine a área externa do polígono, ou seja, a área das "bordas".

(e) Determine quantas linhas seriam produzidas se todos os vértices (cantos) do polígono fossem ligados com os outros vértices. Por exemplo, para os polígonos da figura teríamos 0, 2 e 5 retas.

4 Vetores/listas

4.0 Escreva um programa que recebe um vetor de 10 números inteiros e depois cumpre as seguintes tarefas:

(a) encontra e imprime o menor e o maior números do vetor.

(b) escreve o vetor na ordem inversa.

(c) calcula a média dos elementos do vetor. Depois mude o seu programa para também imprimir o elemento do vetor que tem o valor mais próximo da média.

(d) examina o vetor para ver se há elementos repetidos, imprimindo-os se houver.

4.1 Escreva um programa que recebe um vetor de 20 elementos e depois descobre se existem 2 elementos dentro dele que somados dão 15.

4.2 Escreva um programa que recebe um vetor de 20 elementos e depois reverte o vetor: se o vetor era $\{1, 2, 3, 4, \ldots, 20\}$ ele passa a ser $\{20, \ldots, 4, 3, 2, 1\}$. Não use outro vetor para ajudar!

4.3 Escreva um programa que recebe um vetor de 20 elementos e depois descobre se existem 2 elementos dentro dele que somados dão um outro elemento do vetor.

4.4 Escreva um programa que recebe um vetor de 20 elementos e depois descobre se existem 3 elementos dentro dele que somados dão um outro elemento do vetor.

4.5 Escreva um programa que recebe um vetor de 20 elementos e depois descobre quais os 3 elementos do vetor que estão um ao lado do outro e tem a maior soma.

4.6 Escreva um programa que recebe uma palavra e diz se a palavra é um palíndromo (um palíndromo é uma palavra que é a mesma de trás para a frente, como "arara").

4.7 Mude seu programa do exercício 4.6 para receber uma frase de várias palavras (use a função *fgets*() para isso) e dizer se a frase é um palíndromo. Cuidado: ao analisar a frase você só deve levar em conta as letras e por isso deve pular os espaços em branco.

4.8 Escreva um programa que recebe um vetor de 20 elementos e depois descobre a maior sequencia de números ímpares um ao lado do outro que estão dentro do vetor.

4.9 Escreva um programa que recebe um vetor (por exemplo [2, 3, 4, 7, 1]) e substitui o elemento da posição i pelo produto de **todos os outros** elementos do vetor. Para o exemplo, o resultado seria [84, 56, 42, 168].

4.10

4.11 Escreva um algoritmo que faz a leitura de 15 números inteiros e coloca-os em um vetor, mas impedindo que valores repetidos entrem no vetor. Seu programa deve terminar apenas quando o vetor estiver cheio.

4.12 Escreva um algoritmo que recebe um vetor de 15 elementos contendo números positivos e negativos e imprime os **primeiros** 4 números positivos no vetor. Os números devem ser impressos na mesma ordem em que entraram no vetor.

4.13 Escreva um algoritmo que recebe um vetor de 15 elementos contendo números positivos e negativos e imprime os **últimos** 4 números positivos no vetor. Os números devem ser impressos na mesma ordem em que entraram no vetor.

4.14 Escreva um algoritmo que recebe dois vetores A e B com 10 e 15 elementos respectivamente e encontra o maior elemento que está contido ao mesmo tempo nos dois vetores, ou imprime uma mensagem se não houver nenhum elemento em comum.

4.15 Escreva um programa que recebe uma palavra com o máximo de 20 caracteres e troca a posição de duplas de caracteres: `caramelo` vai ser transformado em `acaremol`. Para ler a palavra use a função `scanf()` com o operador `%20s`.

4.16 Escreva um programa que recebe uma string de caracteres e imprime as palavras da string uma a uma.

4.17 Escreva um programa que recebe uma string de até 20 caracteres e verifica se ela é uma string **legal**. Uma string legal deve seguir as regras:

- Ela não tem "cd", "pq", "xy" ou "uv" ou "vu".
- Ela tem pelo menos três vogais.
- Ela tem pelo menos uma letra duplicada; "aa", "bb", "cc", etc.

Assim, abdde não é legal (só tem duas vogais), abddebei é legal. Depois de fazer isso mude seu programa para informar qual/quais regras foram quebradas se a string não for legal.

4.18

4.19 Escreva um programa que recebe uma string de parênteses como **"((())()())(()())"** e diz se os parênteses estão abrindo e fechando corretamente.

4.20 Escreva um programa que recebe uma string de letras que representam uma molécula complexa como **"aDfrfHhFReGhTYebvVBt"**. Depois disso seu programa deve simplificar a molécula: letras que são vizinhas e estão em versão maiúscula e minúscula (como **Hh** ou **vV**) devem desaparecer. Faça essa simplificação o máximo de vezes possível e apresente a molécula final.

4.21 Escreva um programa que recebe uma string de caracteres e imprime a frase com as palavras na ordem certa mas escritas de trás para a frente: **"Vida longa e prosperidade"** se transforma em **"adiV agnol e edadirepsorp"**.

4.22 Escreva um programa que recebe uma string de caracteres e reordena alfabeticamente as letras de cada palavra: **"vida longa e prosperidade"** se transforma em **"adiv aglno e addeeiopprrs"**.

4.23 Escreva um programa que recebe uma linha de texto e a codifica trocando cada letra pela letra seguinte no alfabeto. (No caso de aparecer um `z`, converta-o para `a`). Tome cuidado para converter letras maiúsculas em outras maiúsculas e minúsculas em minúsculas.

4.24 Você precisa escrever um programa, que examina valores de temperatura em um sofisticado datacenter onde o ar condicionado está desregulado. Seu programa deve receber 100000 valores de temperatura obtidos dos sensores e que vão variar entre -11 e 22 graus Celsius, sempre inteiros. Depois de receber os valores seu programa deve informar a média das temperaturas, se existem mais temperaturas positivas ou negativas e imprimir uma lista das temperaturas informando quantas vezes cada uma delas aparece.

4.25

4.26 Usando um vetor com seu nome, escreva um programa que varre o vetor e remove as vogais. Imprima seu novo nome neste formato condensado, e veja se dá para entender. Note que **remover** significa retirar completamente, não quer dizer que você pode substituí-las por espaços em branco, por exemplo.

4.27 Escreva um programa que recebe um vetor como o que está abaixo

```
1 2 5 1 3 1
```

e produz o gráfico de barras descendentes a seguir:

```
* * * * * *
  * *   *
    *   *
    *
    *
```

4.28 Escreva o programa que decodifica os textos codificados pelo programa feito em 4.23.

4.29 Ligue os elementos de mesmo valor nos dois vetores abaixo:

```
1 4 3 5 6 7 2
2 4 5 3 7 1 6
```

Se você ligou os elementos, então as linhas devem ter se cruzado um certo número de vezes. Escreva um algoritmo que recebe os dois vetores como mostrado acima e calcula o número de cruzamentos das linhas.

4.30 Crie um programa que preenche um vetor V com trinta inteiros com valores entre zero e cinquenta. Em seguida, implemente duas outras funções em seu programa:

(a) Uma função que examina o vetor e imprime os elementos repetidos;

(b) Uma função que examina o vetor e imprime os elementos que não se repetem.

Experimente seu programa, imprimindo o vetor original e comparando com os resultados das duas funções. Verifique se está tudo ok.

4.31 Escreva um programa que recebe números inteiros e armazena-os no vetor A. Em seguida distribua os elementos de A entre outros dois vetores, B e C, colocando os elementos pares em B e os ímpares em C. Depois imprima A, B e C e verifique que está tudo certo.

4.32 Escreva um programa que recebe um vetor como o que está abaixo

```
1 2 1 5 3 1
```

e produz o gráfico de barras ascendentes a seguir:

```
      *
      *
      * *
  *   * *
* * * * * *
```

4.33 Se você acha que o codificador do exercício 4.23 é muito primitivo, pode fazer um mais avançado: em vez de substituir cada caracter pelo caracter que está na posição seguinte, faça um algoritmo que se adapta: a primeira letra é trocada pela seguinte no alfabeto, mas a segunda é trocada pela letra que está duas posições depois, a terceira pela letra três posições depois, e assim por diante. Desta forma, a mensagem **"socorro"** é codificada para **"tqfswxv"**.

4.34 Escreva o programa que decodifica mensagens criadas pelo programa feito em 4.33. Tente não arrancar muitos cabelos.

4.35 Escreva um programa que imprime as primeiras linhas do triângulo de Pascal. Calcule e imprima as primeiras quinze linhas uma depois da outra, mas para fazer isso você só deve usar um único vetor de vinte inteiros. Desta vez não use funções e tente ser o mais veloz possível.

4.36 Escreva um algoritmo que recebe um vetor `S` de 100 caracteres que representa uma linha de texto contendo palavras separadas por espaços em branco, e imprime as palavras dessa linha, uma de cada vez. Na saída, não deixe linhas em branco.

4.37 Escreva um algoritmo que recebe um vetor A de valores inteiros e reorganiza A de forma que os valores pares estejam nas posições iniciais de A e os valores ímpares sejam colocados no final. Você não deve usar outros vetores para ajudar na tarefa!

4.38

4.39 De aorcdo com uma pqsieusa de uma uinrvesriddae ignlsea, não ipomtra em qaul odrem as lrteas de uma plravaa etãso, a úncia csioa iprotmatne é que a piremria e útmlia lrteas etejasm no lgaur crteo. O rseto pdoe ser uma ttaol bçguana que vcoê pdoe anida ler sem mituo pobrlmea. Itso é poqrue nós não lmeos cdaa lrtea isladoa, mas a plravaa cmoo um tdoo.

Para aproveitar este efeito, escreva um algoritmo que recebe uma linha de texto e faz a transformação descrita acima, ou seja, troca algumas letras de posição **dentro** de cada palavra, mas deixa as letras do início e do fim no lugar certo. Em seguida adapte seu algoritmo para transformar um texto inteiro, palavra por palavra.

4.40 Você está usando vetores para representar conjuntos de valores. Agora você deseja programar métodos para fazer as operações mais comuns entre conjuntos:

(a) união de dois conjuntos A e B

(b) intersecção de dois conjuntos A e B

(c) diferença de dois conjuntos A e B

Escreva os algoritmos para fazer estas operações. Pense que A e B podem ter tamanhos diferentes.

4.41 Escreva um programa que recebe um vetor de números inteiros e verifica se algum número que está no vetor é a soma de outros dois números do vetor. Como seu programa se comporta se um dos elementos do vetor for zero?

4.42 Escreva o algoritmo que recebe duas palavras a e b e dá um destes três diagnósticos:

(a) as palavras a e b são iguais

(b) as palavras a e b tem o mesmo tamanho, mas alguns caracteres são diferentes (dizer quantos)

(c) a palavra a inclui a palavra b (dizer quantas vezes)

(d) a palavra b inclui a palavra a (dizer quantas vezes)

4.43 Escreva um algoritmo que recebe 30 palavras e em seguida imprime as palavras fazendo-as caber em linhas de 80 caracteres, indo até a margem direita. Seu algoritmo deve escolher que palavras ficarão em que linhas e adicionar brancos entre as palavras para que as linhas sempre terminem na posição 80.

5 Matrizes

5.0 Escreva um algoritmo que imprime uma matriz de inteiros de tamanho 15×15 em duas versões:

(a) Toda a matriz é impressa;

(b) Só os elementos diferentes de zero são impressos e os zeros saem como espaços em branco.

5.1 Declare uma matriz de caracteres $M_{80\times24}$. Sobre esta matriz, faça as seguintes operações:

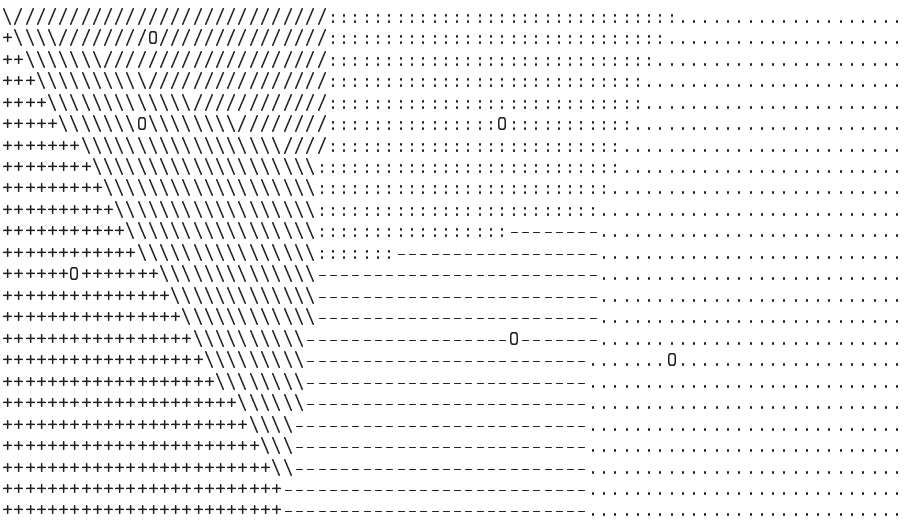
(a) Inicialize todos os elementos com espaços em branco.

(b) Usando a função do exercício 10.1, escolha cinco posições quaisquer da matriz e coloque um caracter '.' nestas posições. Estes pontos serão chamados de sementes.

(c) Para as posições m_{ij} que ainda estão em branco, descubra qual a sua distância até a semente S_{kl} mais próxima. A fórmula de distância a usar pode ser a do exercício 1.16, mas você também pode usar a variação abaixo (ou outra que você deseje):

$$d(m_{ij}, S_{kl}) = \max\left(|i-k|, |j-l|\right)$$

(d) Para cada caracter na matriz, coloque o número da semente que está mais próxima (de '1' a '5', ou use outros caracteres se desejar), e imprima a matriz final. Se desejar, tente com outras fórmulas de distância, mais sementes ou matrizes maiores. Em um exemplo usando caracteres diferentes dos sugeridos acima, obtivemos o seguinte resultado:



5.2 Usando a mesma matriz de caracteres do exercício 5.1, inicialize-a com brancos e em seguida passe apenas um elemento m_{ij} de cada coluna j para '`O`'. O elemento a ser alterado está na linha i, dada por uma das fórmulas abaixo (invente outras depois de testar estas):

$$\begin{aligned} i(j) &= 12 + 11 * \sin(j/4.0) \\ i(j) &= 2 + 24 * (j-40) * (j-40)/2000 \\ i(j) &= 12 + j * \sin(j/6.0)/9 \end{aligned}$$

Tome cuidado com problemas de conversão entre inteiros e números de ponto flutuante!

5.3 Usando a mesma matriz de caracteres de 5.1 (ou maior, se desejar), escreva um programa que a inicializa com brancos e depois coloca 200 elementos '`x`' que servem como sementes em posições aleatórias. Em seguida seu programa aplica as regras simultaneamente (o que isso significa?) sobre **todos** os elementos da matriz:

(a) Se um elemento tem três vizinhos que são sementes, ele vira semente.

(b) Se um elemento tem menos de dois vizinhos que são sementes, ele vira branco.

(c) Se um elemento tem quatro ou mais vizinhos que são sementes, ele vira branco.

Aplique estas regras várias vezes sobre toda a matriz, imprimindo a matriz a cada vez e verificando como ela evolui. Tente alterar as regras ou mudar o número de sementes e reinicie o processo. **Depois** de implementar este programa e ver como ele funciona, acesse o site `http://www.math.com/students/wonders/life/life.html` e experimente uma versão maior...

5.4 Descubra como funciona o jogo 2048 (`http://gabrielecirulli.github.io/2048/`) e programe-o. Depois tente mudar as regras para ver se fica mais fácil.

6 Funções

6.0 Escreva uma função que recebe três inteiros a, b e c e retorna o maior dos três inteiros.

6.1 Escreva uma função que recebe um inteiro n e retorna $n/2$. Depois escreva uma função que recebe um inteiro n e retorna $3n+1$. Escreva uma nova função que usa as funções que você já tem. A nova função recebe um número n e se ele for ímpar retorna $3n+1$, se for par retorna $n/2$.

6.2 Escreva um programa que recebe um inteiro e usa a função do exercício anterior repetidas vezes até que o resultado seja 1, imprimindo os números que vão sendo obtidos.

6.3 Adapte o programa do exercício anterior para descobrir qual número entre 1 e 100000 produz a sequencia mais longa até chegar em 1. Imprimir os números que vão sendo obtidos pode não ser uma boa ideia agora...

6.4 Escreva uma função que recebe um vetor de caracteres (terminado por '\0') e devolve o número de vogais que estão no vetor.

6.5 Escreva uma função `int ehNumero( char vet[] )` que recebe um vetor de caracteres terminado por '\0' e retorna 0 ou 1 se o vetor pode estar representando um número (pense que um número só pode ter os dígitos de 0 a 9 e mais um único ponto decimal).

6.6 Escreva uma função `float valor( char vet[] )` que recebe um vetor de caracteres terminado por '\0' que representa um número (como por exemplo "3.14159") e retorna um float com o mesmo valor.

6.7 Escreva uma função `void contaTriplo( char vet[], int *vog, int *cons, int *simb)` que recebe um vetor de caracteres (terminado por '\0') e deixa escrito em `vog`, `cons` e `simb` o número de vogais, consoantes e símbolos (não-vogais nem consoantes) que estão no vetor.

6.8 Escreva uma função `void viraFrase( char vet[] )` que recebe um vetor de caracteres (terminado por '\0') e inverte o vetor: "Papagaio azul e verde" deve ser transformado em "edrev e luza oiagapaP".

6.9 Escreva uma função `void mataVogais( char vet[] )` que recebe um vetor de caracteres (terminado por '\0') e retira as vogais do vetor: "Papagaio azul e verde" deve ser transformado em "Ppg zl vrd".

6.10 Escreva uma função `void viraPalavras( char vet[] )` que recebe um vetor de caracteres (terminado por '\0') e inverte as palavras dentro do vetor: "Papagaio azul e verde" deve ser transformado em "oiagapaP luza e edrev".

6.11 Escreva uma função que recebe um inteiro n e devolve o fatorial de n.

6.12 Use a função fatorial do exercício anterior para escrever a função `comb(int n, int k)`, que retorna o número de combinações de n elementos escolhidos em grupos de k elementos.

6.13 Faça uma função que recebe uma matriz $A_{5\times5}$ e dois inteiros i e j, trocando as linhas i e j da matriz. Imprima o resultado e confirme.

6.14 Faça uma função que recebe uma matriz $A_{10\times10}$ e a transpõe, ou seja, o elemento que está em $A_{2,3}$ vai para $A_{3,2}$ e vice-versa.

6.15 Escreva uma função que recebe um inteiro n e escreve n em binário. Depois altere a função para escrever n em uma base b que também é enviada para a função.

6.16 Escreva uma função que recebe inteiros m e n e determina o maior divisor comum de m e n. Sugestão: use o algoritmo de Euclides que está em 3.22 para isso.

6.17 Escreva um algoritmo que controla a compra de entradas em um cinema. O algoritmo deve representar a ocupação dos assentos da platéia (40 filas de 50 assentos cada), onde cada assento pode estar reservado (marcado com `R`), comprado (marcado com `C`), ou livre (marcado com `.`). Você deve fornecer uma função que procura um lugar livre na platéia e informa sua posição, bem como outra função que imprime um mapa da platéia, marcando os lugares de acordo com sua situação, imprimindo em seguida os totais de lugares ocupados, reservados e vagos.

6.18 Escreva o algoritmo de uma função que recebe as informações sobre o cinema da questão anterior e acha dois lugares livres para um casal.

6.19 Escreva uma função que recebe as informações sobre o cinema da questão anterior e acha um bloco retangular de lugares livres para uma turma de escola. Você pode apresentar dois algoritmos diferentes dependendo do como o pedido pode ser feito:

(a) Assuma que a turma precisa de um bloco retangular de $m \times n$ lugares.

(b) Assuma que a turma precisa colocar n alunos em bloco, mas não importam as dimensões do bloco.

6.20 Escreva uma função que recebe um inteiro n e informa quantos bits são 1 quando n é escrito em binário.

6.21 Descubra o que faz a função abaixo. Para testar, use números positivos.

```
int mistery( int a, int b ) {
   int r = 0;
   while ( b > 0 ) {
     r += (b % 2) * a;
     b /= 2;
     a *= 2;
   }
   return r;
}
```

6.22 Escreva uma função que recebe um inteiro n e retorna o número reverso de n. Por exemplo, se a função receber o número 123 ela deve retornar o inteiro 321. Sua função não deve dar certo apenas com números de três dígitos, ela deve ser genérica.

6.23 Escreva um método `pow(int x, int n)` que recebe `x` e `n` e calcula x^n. Seu método também deve funcionar corretamente quando $n = 0$ ou $n = 1$. Obviamente você não deve usar a função que já existe na sua linguagem de programação.

6.24 Escreva uma função `numdiv(int n)` que recebe um inteiro n e retorna o número de divisores de n. Depois escreva uma função `ehprimo(int n)` que verifica se n é primo e imprime uma mensagem na tela dizendo se é ou não é.

6.25 Adapte a função `ehprimo(int n)` para ter um programa que recebe um inteiro n e encontra o primeiro primo maior ou igual a n.

6.26 Escreva uma função que determina raizes quadradas, usando o método do problema 3.27. Esta função recebe um número x e retorna (evidentemente) $\sqrt{x}$.

6.27 Escreva uma função `char * copy(char * s)` que recebe uma string `s` e faz uma cópia de `s` usando `malloc` para alocar espaço, copiando cada caratere e retornando um apontador para a cópia.

6.28 Escreva uma função que calcula o valor (aproximado) de $\cos(x)$, dado pela aproximação abaixo:

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

(a) Faça uma função que faz a soma com três termos. Escreva um programa que recebe valores de x e calcula seu cosseno com esta função, depois calcula usando a função cosseno disponível na sua linguagem preferida e apresentando a diferença entre os resultados obtidos. Você pode achar útil escrever funções separadas para calcular potências e para calcular fatoriais.

(b) Adapte sua função para receber x e também o número n de termos que devem ser somados, assim o usuário pode controlar a precisão do cálculo.

6.29 Ninguém sabe quem propôs a conjectura dos primos gêmeos, mas ela sugere que existe um número infinito de pares de números primos na forma k, $k+2$. Por exemplo, 17 e 19 são primos gêmeos, assim como 227 e 229 e muitos (infinitos?) outros pares. Usando o conhecimento do exercício 3.14, escreva um programa que acha pares de primos gêmeos.

6.30 Use seu conhecimento adquirido nos exercícios anteriores para imprimir as primeiras doze linhas do triângulo de Pascal como mostrado abaixo. Faça isso usando funções e não se importe com o tempo gasto.

```
1  1
1  2   1
1  3   3   1
1  4   6   4  1
1  5  10  10  5  1
```

7 Structs

7.0 Crie uma `struct` chamada `fraction` que representa uma fração. Você pode escolher entre duas alternativas:

(a) Apenas com numerador e denominador. Neste caso o sinal da fração fica sempre com o numerador

(b) Com numerador, denominador e sinal da fração. Neste caso, o numerador e o denominador são valores sempre positivos.

Depois de escolher como quer fazer sua `struct`, defina também as funções que operam sobre ela:

(a) `struct fraction mult(struct fraction a, struct fraction b);`

(b) `struct fraction add(struct fraction a, struct fraction b);`

(c) `struct fraction reduce(struct fraction a);`

O funcionamento das duas primeiras funções deve ser evidente. A terceira função recebe uma fração e a reduz até que numerador e denominador não tenham divisores em comum. (Para isso, você deve achar útil ter feito o exercício 3.22). Depois de programar as três funções, teste-as e tenha certeza de que elas funcionam corretamente. Existe alguma diferença significativa em guardar-se o sinal separadamente ou isso não tem importância?

7.1 Use as frações do problema 7.0 e reprograme o exercício 3.27, fazendo as contas com as frações que você definiu. Com isto você não deve mais achar `float`s que chegam perto da raiz de um número, mas deve achar frações que se aproximam... ache frações que se aproximam de $\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}$ e confirme o resultado! Já esteja alerta para o fato de que os numeradores e denominadores crescem rapidamente e logo estouram a capacidade de armazenamento dos números inteiros, permitindo apenas quatro ou cinco iterações.

7.2 Apresente uma struct para guardar uma data (dia, mês, ano). Depois cumpra as tarefas abaixo:

(a) escreva uma função que imprime uma data com o formato adequado;

(b) escreva uma função que recebe duas datas e imprime a data mais antiga;

7.3 Planeje structs para armazenar os dados de ligações telefônicas e os dados de usuários de telefones. Depois faça o algoritmo para listar todas as chamadas feitas por um usuário em um mês de uso do telefone. Não esqueça de justificar quaisquer decisões importantes que você tenha tomado ao planejar os registros.

7.4 Crie uma struct que representa os dados de um terreno retangular. Depois de escolher como representar o terreno, declare um vetor de 100 terrenos e apresente algoritmos para:

(a) Listar os terrenos que são quadrados.

(b) Encontrar o terreno de maior perímetro.

(c) Encontrar o terreno que mais se assemelha a um quadrado.

(d) Encontrar o terreno que menos se assemelha a um quadrado.

7.5 Crie uma `struct` que representa um planeta pertencente à Federação, contendo:

(a) Nome do planeta, raio (em km), massa (em kg).

(b) Número de satélites, número de habitantes.

(c) Quadrante (de 0 a 8) e setor da galáxia (de A a L).

7.6 Usando os dados do exercício anterior, crie vinte planetas e apresente algoritmos para:

(a) Determinar o número de habitantes no quadrante Gamma (o código para o quadrante Gamma é 6).

(b) Determinar qual o setor mais populoso da galáxia.

(c) Encontrar o maior planeta de cada quadrante.

7.7 Crie uma `struct` que representa uma nave da Federação, contendo:

(a) Nome da nave, classe e tamanho (toneladas).

(b) Número de tripulantes e de naves auxiliares.

(c) Planeta de origem e planeta de destino.

(d) Quadrante e setor atuais na galáxia.

7.8 Escreva um programa que cria vinte planetas e dezesseis naves. Preencha os dados das naves e planetas com o gerador de nomes 10.2 feito anteriormente, e coloque números ao acaso para os valores adequados. Depois de tudo isso seu programa deve pedir um quadrante e um setor da galáxia, imprimindo um relatório com os dados dos planetas e das naves no local.

7.9 Coloque doze naves romulanas em seu programa. Emita um relatório se naves romulanas e naves da Federação estiverem na mesma posição. No relatório, diga quais naves se encontraram e qual delas foi destruída. Faça com que sobre apenas uma nave em cada quadrante e tome cuidado pois podem haver **mais** de duas naves.

7.10 Adapte seu programa para pedir o nome de uma nave e de um planeta. Redirecione a nave para o planeta, mas nao esqueça de verificar se a nave e o planeta realmente existem! Verifique se há romulanos outra vez e deixe outra batalha acontecer.

7.11 Crie portões espaciais: inclua uma opção na qual é pedido o nome de uma nave. Esta nave entra em um portão e sai em outro lugar da galáxia. Seu programa deve dizer a posição antiga e a nova, verificando os romulanos outra vez. Não esqueça de verificar se você caiu no mesmo lugar de onde saiu, isto não pode acontecer!

7.12 Crie uma `struct` que representa os dados de um submarino nuclear:

(a) Nome e classe.

(b) Deslocamento (toneladas) e tripulantes.

(c) Número de torpedos (entre 10 e 16).

(d) Posição e direção (dois inteiros para cada um)

Coloque doze submarinos do time A e doze do time B em um tabuleiro de 12×12 posições tomando cuidado para que não haja dois submarinos na mesma posição. Em seguida deixe que eles se destruam em rodadas: cada submarino que ainda tem munição atira um torpedo que atinge alguma outra posição do tabuleiro, aniquilando o que houver lá (mesmo que seja do mesmo time). Depois de atirar os submarinos se movem na direção adequada. Em caso de destruição, um relatório é emitido. Quando ninguém mais tiver torpedos, emita um relatório com a situação final.

7.13 Altere o jogo do exercício 7.12 mostrando na tela o tabuleiro e confirmando visualmente o que está acontecendo.

8 Arquivos

8.0 Escreva um programa que lê inteiros de um arquivo e encontra o maior inteiro no arquivo. Perceba que quando abrir o arquivo você não sabe quantos inteiros ele contém.

8.1 Escreva um programa que lê um arquivo contendo inteiros e escreve dois arquivos, um contendo os números ímpares e outro contendo os números pares.

8.2 Escreva um programa que lê inteiros de um arquivo e encontra o maior inteiro no arquivo e quantas vezes ele aparece.

8.3 Escreva um programa que lê palavras de um arquivo e depois informa quantas palavras eram de tamanho 1, quantas palavras eram de tamanho 2, e assim por diante até tamanho 20.

8.4 Escreva um programa que lê dois arquivos A e B e informa se eles são exatamente iguais ou (se não forem) informa o primeiro caractere em que eles ficaram diferentes. Informe também em que linha dos arquivos isso aconteceu.

8.5 Escreva um programa que lê dois arquivos A e B e escreve um arquivo C, onde é colocada uma letra de A e depois uma de B e assim por diante. Então depois de ler A="aboboras" e B="caramelo" o arquivo C será "acbaorbaomrealso". Faça isso até que o arquivo mais curto termine.

8.6 Faça um programa que lê um arquivo produzido pelo exercício 8.5 e traz de volta os dois arquivos iniciais.

9 Recursão

9.0 Escreva um método `soma(int i, int j)` que recebe `i` e `j` e retorna a soma de todos os números de `i` até `j`, **mas faz isso dividindo o trecho em duas partes e depois soma cada parte separadamente.**

9.1 Escreva um método recursivo `fatorial(int n)` que calcula o fatorial de `n` e produz a saída abaixo para o caso `n`=5:

```
Fatorial de 5
Fatorial de 4
Fatorial de 3
Fatorial de 2
Fatorial de 1
Fatorial de 0
Retorna 1
Retorna 1
Retorna 2
Retorna 6
Retorna 24
Retorna 120
```

9.2 Altere o método recursivo `fatorial(int n)` que calcula o fatorial de `n` para produzir **exatamente** a saída abaixo para o caso `n`=5:

```
Fatorial de 5
   Fatorial de 4
      Fatorial de 3
         Fatorial de 2
            Fatorial de 1
               Fatorial de 0
               Retorna 1
            Retorna 1
         Retorna 2
      Retorna 6
   Retorna 24
Retorna 120
```

9.3 Escreva um método recursivo `numerodeuns(int n)` que recebe um número `n` e retorna quantos digitos 1 existem na representação binária de `n`.

9.4 A conjectura de Goldbach foi apresentada por Christian Goldbach em uma carta a Leonhard Euler no ano de 1742, e diz:

> *Todo número par maior do que dois pode ser representado pela soma de dois números primos.*

Apesar de até hoje não ter sido provada, a conjectura funcionou para todos os casos que foram experimentados. Adapte o programa feito no exercício 3.14, fazendo uma função que recebe um número n e retorna zero ou um para indicar se n é primo. Em seguida, use esta função em um programa maior, que recebe um número par m e tenta comprovar a conjectura para m.

9.5 Uma matriz $M_{256\times256}$ pode ser dividida em 4 sub-matrizes quadradas:

I	II
III	IV

Escreva o algoritmo que troca os elementos que estão nas sub-matrizes II e III. Em seguida, aplique o algoritmo recursivamente para visitar as quatro sub-matrizes, fazendo a mesma divisão e troca em cada uma delas. Seu algoritmo só deve parar quando a sub-matriz tiver apenas um elemento. Invente uma forma de apresentar a matriz ao final da sequencia de operações (saída gráfica, caracteres, etc.).

10 Números aleatórios

Para usar números aleatórios em C você deve preparar o terreno:

- Incluir alguns headers:

```
#include<stdlib.h>
#include<time.h>
```

- Chamar uma vez a função que prepara números aleatórios, antes de usá-los:

```
srand ( time (NULL) );
```

- Depois é só usar a função adequada:

```
int meunum = rand(); // Retorna um inteiro que vai de 0 até RAND_MAX
```

10.0 Escreva uma função que vai escolher faixas aleatoriamente do seu player de MP3. Você quer escolher um número de faixa entre 1 e n e não quer que seja um número igual aos dois últimos números sorteados.

10.1 Escreva uma função que recebe um inteiro n e a cada vez que for chamada retorna um valor inteiro entre 0 e $n-1$. (Não é difícil inventar uma função assim, mas é excepcionalmente difícil fazer uma **boa** função onde os números não fiquem se repetindo de forma óbvia.) Se você quiser mais detalhes, sugere-se uma olhada na Wikipedia.

10.2 Escreva um gerador de nomes exóticos. Crie um vetor de sílabas, como 'elp', 'tram', 'lat', 'rem', 'thor', 'vax', 'loy', 'fen', 'nax' e quaisquer outras que você ache interessantes (quanto mais, melhor!). Em seguida gere palavras ao acaso, com tamanhos diferentes (sorteie o número de sílabas). Escolha as mais interessantes e compare com as de seus colegas. Se quiser ser realmente sofisticado, crie vetores para sílabas iniciais, sílabas de meio e sílabas finais de palavras, assim você pode escolher onde elas vão ficar.

11 Arquivos

11.0 Usando seu conhecimento de números aleatórios (seção 10), escreva um programa que recebe um número inteiro n e um nome de arquivo e escreve n inteiros aleatórios em um arquivo com o nome que foi dado. Os números devem variar entre 0 e 1000.

11.1 Escreva um programa que lê o arquivo de números aleatórios e descobre quantos números pares e ímpares estão no arquivo. Lembre que você **não sabe** o total de números que estão no arquivo!

11.2 Escreva um programa que lê o arquivo de números aleatórios e verifica quantas vezes acontece de dois números que estão juntos no arquivo somam exatamente o número que vem depois deles. Lembre que você **não sabe** quantos números estão no arquivo!

11.3 Escreva um programa que recebe o nome de um arquivo A que já existe e o nome de outro arquivo B e faz uma cópia de tudo que tem em A para B, destruindo B se já tiver alguma coisa lá.

12 Problemas extras

Alguns dos problemas que estão nesta seção podem ter soluções bastante lentas se usarem apenas laços e mais nenhum armazenamento. Muitas vezes existem soluções muito mais eficientes, portanto vários destes exercícios podem ser feitos **duas vezes,** uma com laços e depois usando-se outras formas de aceleração.

12.1 Vendo virado

Em uma calculadora, quando olhamos com o visor de cabeça para baixo alguns números ainda podem ser lidos (só que com outros valores). Por exemplo, 1995 vira 5661. O quinto número que pode ser lido de cabeça para baixo é 8, e o décimo-quinto é 21, que passa a valer 12 quando é visto de cabeça para baixo. Você deve responder uma pergunta básica:

- Qual o milionésimo número que pode ser reconhecido como outro número, quando é visto de cabeça para baixo?

12.2 Fazendo continhas

Você deve resolver um problema simples: para um número inteiro n, um **sequenciamento** de n é uma sequencia de números consecutivos $i, i+1, i+2, \ldots, i+j$ onde a soma de todos estes números é igual a n. Traduzindo temos

$$n = i + (i+1) + (i+2) + (i+3) + \cdots + (i+j).$$

Depois de saber disso, você pode verificar por si mesmo que muitos números tem mais de um sequenciamento. Por exemplo, o número 15 tem quatro sequenciamentos (de comprimentos 5, 3, 2 e 1):

$$\begin{aligned} 15 &= 1+2+3+4+5 \\ &= 4+5+6 \\ &= 7+8 \\ &= 15 \end{aligned}$$

Agora, com todas estas importantes informações, você deve resolver um problema bastante simples: descobrir qual o número entre um e um milhão tem mais sequenciamentos.

12.3 Os dez milhões de pedacinhos

Você deve determinar qual número entre 1.000.000 e 1.500.000 pode ser representado mais vezes como o produto de três inteiros positivos maiores do que um (não necessariamente diferentes). Na contabilização a ordem dos inteiros não é importante. Assim, $a * b * c$ equivale a $b * a * c$. Por exemplo, nestas condições existem 114 triplas de números que quando são multiplicados resultam em 1.000.000.

12.4 Os números de Frodo

Os números de Frodo são definidos de forma bastante simples: o número de Frodo $F(n)$ para $n > 1$, é o menor número positivo que pode ser dividido por todos os valores de 1 a n. Conhecendo esta definição, seu trabalho é determinar um algoritmo para achar os números de Frodo e apresentar uma tabela destes números até $n = 40$. Você também deve levar em conta que os números crescem depressa. Para $n > 22$, será necessário usar números de 64 bits para representar o resultado correto.

12.5 Fabricando números

Seja dado um número $a = 1$. Começando com a, outros números podem ser produzidos seguindo-se as duas regras abaixo:

12.0 Multiplicando por 2 um número já produzido.

12.1 Dividindo por 3 um número já produzido e descartando a parte fracionária (ou seja, fazendo uma divisão inteira).

Por exemplo, seguindo sucessivamente estas regras podemos ter

$$10 = 1 \times 2 \times 2 \times 2 \times 2 \times 2 \div 3.$$

Seu problema é produzir todos os inteiros de 2 a 100 seguindo apenas as regras acima. Para a saída, todos os números devem ser apresentados em ordem crescente, apresentando-se quantas vezes tivemos de usar cada regra. Por exemplo, a linha da saída para 10 seria a seguinte:

```
10 = 2^5 / 3^1
```

Os resultados possíveis não são únicos, todos os inteiros podem ser obtidos em mais de uma forma. Por exemplo, 3 e 40 podem ser obtidos pelo menos nas duas formas abaixo:

$$\begin{array}{rcccc} 3 & = & 2^5 \div 3^2 & = & 2^{119} \div 3^{74} \\ 40 & = & 2^{64} \div 3^{37} & = & 2^{83} \div 3^{39}, \end{array}$$

portanto deve-se dar preferência aos valores menores. Algumas regras e considerações:

12.2 É permitido usar inteiros especiais, como `long`, `BigInt` e outras implementações de inteiros extra-longos.

12.3 Não é permitido usar `float`, `real` e `double`. Funções como `pow(x,y)` também são **inválidas**.

12.4 Se um número não precisa ser dividido por 3, pode-se apresentar `3^0` na saída.

12.6 Você e as megacorporações!

Os diretores de uma grande corporação resolveram promover uma nova política de estímulo aos funcionários. Esta política resume-se em dar aos funcionários alguns minutos por dia para respirar, tomar café, apontar os lápis e colocar fofocas em dia. Infelizmente, há problemas para decidir exatamente os horários em que isso deve ser feito. Um dos gerentes dá a seguinte sugestão: "Podemos usar um relógio de três ponteiros (horas, minutos e segundos) e cada vez que os três ponteiros formarem uma estrela (formada por três pontas a 120 graus uma da outra: ), teremos um desses minutos de concentração! Eles ficarão espalhados pelo dia e teremos minutos de folga!!" Claro que o gerente é um idiota que deseja impressionar, mas a diretoria engoliu e sobrou pra você resolver o resto do problema.

É importante que você conheça os relógios da fábrica para saber como resolver o problema:

- O ponteiro dos segundos **não** se move continuamente entre as marcações, mas dá saltos a cada segundo;
- O ponteiro das horas move-se também aos saltos entre as marcações de minutos.
- O ponteiro dos minutos também se move em saltos entre os minutos.

Você deve fornecer as seguintes informações:

12.5 O total de vezes em que estes minutos de descanso acontecem em um dia;

12.6 Uma listagem deles no formato `hh:mm:ss`.

12.7 O maior intervalo encontrado entre dois destes minutos.

Não esqueça de que as estrelas podem ser encontradas em **qualquer** posição, podendo estar rotacionadas, de cabeça para baixo, etc. Também serão criados os super-minutos, quando a estrela está exatamente em posição vertical. Nestes casos a folga é de cinco minutos. Para agradar a gerência, determine também quantas destas folgas existem em um dia.

12.7 Você e os bancos

Depois de ir num caixa automático e receber vinte reais em moedinhas de 10 e 25 centavos, você se irrita com os bancos que deixam isso acontecer e resolve escrever um programa para calcular sempre o menor número de moedinhas para fornecer uma certa quantia. (Seu plano é vender o programa para os bancos depois, por uma fortuna.). Com esta intenção, seu programa deve ler os dados de entrada abaixo:

12.8 O número n de valores de moedas existentes.

12.9 Os valores $a_1, a_2, \ldots, a_n$ destas moedas em centavos. Isto é necessário para que você possa exportar seu programa para vários países com moedas diferentes a preços exorbitantes.

12.10 A quantia m a ser fornecida com estas moedas.

Seu programa deve ler estes dados e determinar o menor número de moedas necessário para formar m, imprimindo este número e quais as moedas usadas.

Por exemplo, se quiséssemos fornecer oitenta e sete centavos com as moedas existentes hoje no Brasil, teríamos a seguinte entrada:

```
6
1
5
10
25
50
100
87
```

Se quiséssemos R$ 1.23 o arquivo conteria a linha `123` ao final e para 87 centavos a saída seria

```
5 moedas
1x50 1x25 1x10 2x1
```

Embora este problema possa parecer simples e descomplicado, sugere-se os seguintes testes básicos:

```
3   3   4   4
1   3   2   3
5   7   3   6
11  12  5   9
15  14  11  10
        17  11
```