1. Soal ini dapat diselesaikan dengan cara mencoba semua kemungkinan garis yang dapat dihapus. Salah satu garis yang dapat dihapus adalah garis (2, 6), seperti yang ditunjukkan di soal.
    - (2, 6). Simpul maksimum yang dapat dipilih dari H: 3, 4, 6
    - (2, 3). Simpul maksimum yang dapat dipilih dari H: 2, 3, 4
    - (3, 5). Simpul maksimum yang dapat dipilih dari H: 3, 4, 5
    - (5, 6). Simpul maksimum yang dapat dipilih dari H: 3, 4, 6
    - (6, 1). Simpul maksimum yang dapat dipilih dari H: 3, 4, 6
    - (5, 1). Simpul maksimum yang dapat dipilih dari H: 3, 4, 6
    - (1, 4). Simpul maksimum yang dapat dipilih dari H: 1, 2, 4

    Dari tujuh kemungkinan di atas, dapat disimpulkan bahwa simpul maksimum yang dapat dipilih dari H adalah 3.

    Jawab: 3

2. ***Di soal ini, terdapat kekurangan informasi. Sebuah pohon N-ary seimbang penuh harus memiliki semua daun di kedalaman yang sama.***

    Berdasarkan fakta bahwa hanya kandang yang merupakan daun yang boleh kosong, kita dapat melihat observasi bahwa setiap kandang yang belum memiliki tepat N sub-pohon pasti hanya memiliki daun sebagai sub-pohon. Maka, hanya jika semua daun telah diisi dengan bebek, barulah kedalaman pohon tersebut dapat ditambah.

    Urutan pengisian bebek pada setiap kandang adalah sebagai berikut:
    - Kedalaman 0 hanya terdiri dari akar pohon dan bisa diisi 1 bebek.
    - Kedalaman 1 terdiri dari N kandang, dan harus diisi penuh dengan N bebek, baru boleh lanjut ke kedalaman berikutnya. Nilai keindahan bertambah sebanyak N.
    - Kedalaman 2 terdiri dari N^2 kandang. Nilai keindahan bertambah sebanyak N^2 * 2.
    - Kedalaman i terdiri dari N^i kandang. Nilai keindahan bertambah sebanyak N^i * i.

    Dengan demikian, kita bisa mencoba-coba N, sehingga kita tahu berapa N maksimum sehingga nilai keindahan masih 50 ke atas.

    Nilai N yang menjadi jawaban adalah 12, karena nilai keindahan pohon 12-ary adalah 50, sedangkan nilai keindahan pohon jika N > 12 akan lebih kecil dari 50.

3. Jika fpb(a, b) = x, dan kpk(a, b) = y, maka a dan b pasti kelipatan x dan faktor y. Maka, kita tahu bahwa ia adalah kelipatan 15 dan faktor 90. Angka yang mungkin untuk a_i hanya 15, 30, 45, dan 90.

   Untuk mempermudah soal ini, kita dapat membagi semua angka dengan 15. Fpb menjadi 1, kpk menjadi 6, dan kemungkinan a_i yang mungkin adalah 1, 2, 3, 6. Karena kemungkinan a_i hanya sedikit, kita dapat mencari semua kemungkinan deret bilangan.

   - 1, 2, 3, 6
   - 1, 2, 6
   - 1, 2, 3
   - 1, 3, 6
   - 1, 6
   - 2, 3, 6
   - 2, 3

   Jawab: 7

4. Berikut adalah langkah-langkah menyelesaikan soal ini.
   - (a→b) = (-a or b).
   - (a or a) = a
   - Notasi (a or -a) pasti bernilai TRUE.
   - Maka, (a or b) or (a→b) = (a or b or -a or b) = (TRUE or b) = TRUE.
   - Sehingga, dapat disimpulkan bahwa setiap himpunan yang mungkin, pasti memiliki 15 nilai TRUE. Nilai maksimum dari banyaknya nilai TRUE dari setiap himpunan juga pasti **15**.

   Jawab: 15

5. Berdasarkan pembahasan soal nomor 4, jawabannya adalah **15**.
   Jawab: 15

6. Kita bisa mencoba satu-satu nilai koin baru dari 2 sampai 30, lalu kita akan cek apakah jika uang bernilai x ditukarkan, jumlah koin yang didapat tetap minimum.
    - Koin dengan nilai 2 dan 3 pasti valid karena nilai tersebut jika dikalikan 2 masih lebih kecil dari nilai koin selanjutnya yang adalah 7.
    - Koin dengan nilai 4 valid, karena ada koin dengan nilai 1. Ini karena 4 + 4 = 1 + 7.
    - Koin dengan nilai 5 dan 6 tidak valid, karena jika nilai x masing-masing adalah 10 dan 12, banyak koin yang didapat tidak minimum, karena penukar uang akan memberikan koin dengan nilai 7 terlebih dahulu.
    - Koin dengan nilai 8, 9, …, 12 tidak valid, karena koin-koin ini nilainya lebih kecil dari dua kali nilai koin sebelumnya yang bernilai 7. Anda dapat mencoba menukarkan uang bernilai x = 14, dan dapat dilihat bahwa dengan menambahkan koin-koin ini, banyak koin yang didapat tidak lagi minimum.
    - Koin dengan nilai 13, bisa valid apabila tidak ada koin 23, sama halnya seperti poin (c). Tetapi karena ada koin dengan nilai 23, koin 13 ini tidak valid (Syarat tidak terpenuhi ketika menukarkan uang dengan nilai 26). Ini juga sama untuk koin dengan nilai 14, 15, …, 22.
    - Koin dengan nilai 24, 25, …, 30 tidak valid, sama halnya seperti poin (c).

Berdasarkan percobaan diatas, banyak nilai koin baru yang mungkin ada **3**, yaitu 2, 3, dan 4.

Jawab: 3

7. Soal ini dapat diselesaikan dengan cara mensimulasikan gerakan robot dengan aturan yang berlaku.

```
#..456#a
..23#789
.X1#..#c
..##.#Yd
#......#
urutan gerakan adalah 1..9, a..e
posisi b bertepatan dengan 9, posisi e bertepatan dengan Y

Total gerakan = 9 (banyaknya angka 1-9) + 5 (banyaknya huruf
```

```
a-e) = 14
```

Jawab: 14

8. Kita dapat mencoba beberapa posisi strategis

```
O menandakan # yang baru dipasang.
#.....#.
....#...
.X.#.B#A
..##.#YO
#......#
langkah yang dibutuhkan untuk mencapai A: 12
langkah yang dibutuhkan untuk mencapai B: 16
total langkah yang dibutuhkan: 22
```

```
#....A#.
....#O..
BX.#..#.
..##.#Y.
#......#
langkah yang dibutuhkan untuk mencapai A: 6
langkah yang dibutuhkan untuk mencapai B: 13
total langkah yang dibutuhkan: 22
```

Dapat dilihat bahwa hanya ada dua posisi strategis saja yang perlu di cek. Untuk posisi strategis pertama, apabila O dipindah sedikit ke atas, hanya akan memperburuk jawaban. Hal yang sama terjadi untuk posisi strategis kedua apabila O digeser ke atas. Penempatan O di posisi selain itu akan memberikan jawaban seperti soal nomor 7.

Maka, jawaban soal ini bisa (4, 7) atau (2, 5).

Jawab: **(4, 7)** atau **(2, 5)**

9. Soal ini dapat diselesaikan dengan menggunakan algoritma **Minimum Spanning Tree (MST)**. Salah satu algoritma MST adalah Kruskal's algorithm. Proses semua jalan dari biaya terkecil, lalu hubungkan jalan apabila jalan tersebut menghubungkan dua buah himpunan desa yang belum pernah terhubung sebelumnya.
    - Jalan (B, C) dengan biaya 5 adalah yang jalan yang pertama kali diambil, menghubungkan himpunan desa {B} dan {C} menjadi {B, C}.
    - Jalan (D, A) dengan biaya 6, dihubungkan dan menghasilkan himpunan {A, D}.
    - Jalan (E, C) dengan biaya 7, dihubungkan dan menghasilkan himpunan {B, C, E}.
    - Jalan (B, E) dengan biaya 7, diabaikan karena B dan E sudah terhubung.
    - Jalan (B, A) dengan biaya 8, dihubungkan dan menghasilkan himpunan {A, B, C, D, E}.

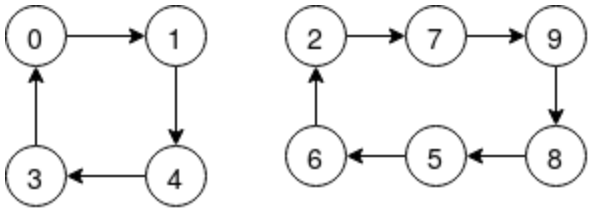
    Dengan begitu, semua desa telah dihubungkan dan biaya total adalah **26**.

    Jawab: 26

10. Kita dapat menggunakan konfigurasi jawaban soal 9 dan mencoba menghapus salah satu jalan yang digunakan di konfigurasi tersebut. Setelah itu, apabila kita melakukan ulang algoritma kruskal, hasil konfigurasi pasti berbeda dengan konfigurasi awal.
    Salah satu konfigurasi yang berbeda dengan soal nomor 9 adalah apabila kita tidak mengambil jalan (E, C), tetapi mengambil jalan (B, E), dengan total biaya **26**.

    Jawab: 26

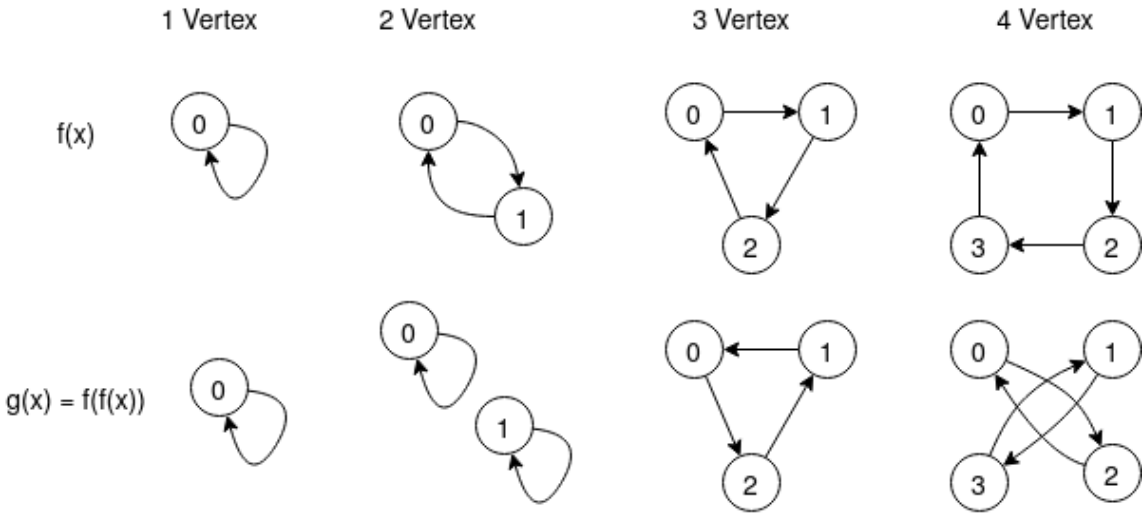
11. Perhatikan bahwa di antara angka hasil pengkodean ini, semua nya unik (tidak ada yang sama). Kita akan membuat graph yang terdiri dari 10 vertex yang berhubungan dengan setiap angka, dan terdapat edge dari angka a ke angka b apabila pengkodean angka a adalah b. Untuk contoh di soal, graph yang akan dibentuk adalah sebagai berikut.

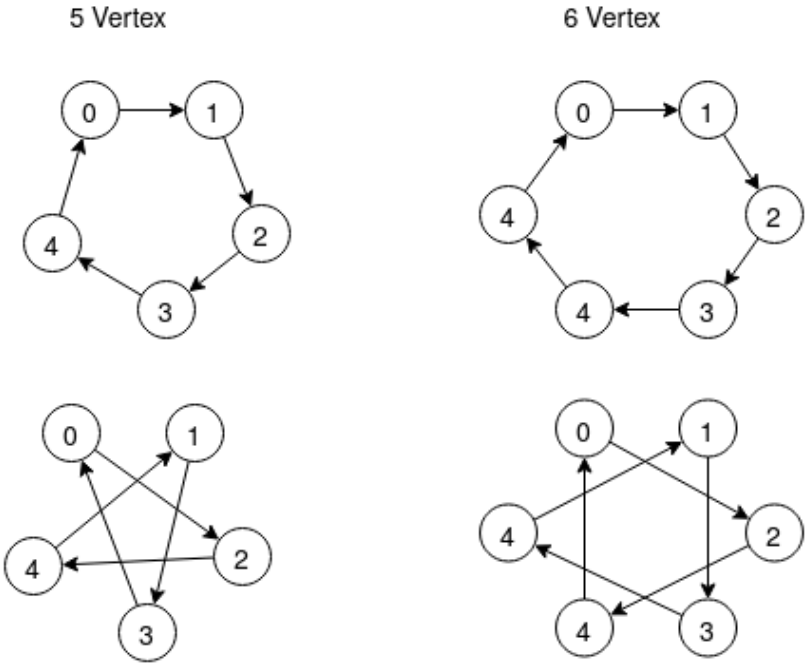


    Karena suatu angka semula hanya akan dikode menjadi satu angka hasil, semua vertex pasti memiliki satu outgoing edge. Karena setiap angka hasil adalah unik, maka semua vertex pasti

memiliki satu ingoing edge. Dengan dua syarat ini, graph yang terbentuk pasti terdiri dari beberapa cycle, dan tidak ada cabang sama sekali.

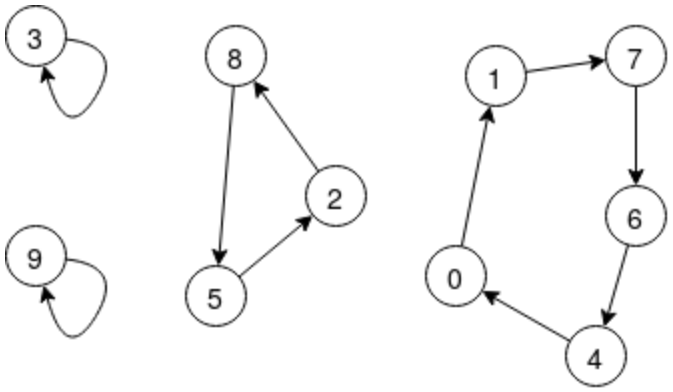
Di soal, diketahui f(f(x)) untuk suatu x, dan dicari f(x). Jika f(f(x)) dijadikan fungsi pengkodean baru g(x), maka graph g(x) dapat dibuat dari graph f(x). Untuk setiap edge (a, b) dan edge (b, c) yang ada di graph f(x), tambahkan edge (a, c) di graph g(x).





Kita dapat melihat pola bahwa setiap cycle yang besarnya ganjil tidak akan berubah besarnya, sedangkan cycle yang besarnya genap akan menjadi dua cycle baru yang besarnya setengah dari semula.

Berikut adalah graph untuk f(f((0123456789)) = 1783024659.



Cycle dengan besar 3 pasti berasal dari cycle dengan besar 3 juga, karena tidak ada cycle lain di gambar diatas yang memiliki besar 3. Maka, f(0123456789) = ??5??8??2?.
Sama halnya dengan cycle dengan besar 5. Maka, f(0123456789) = 645?78102?.
Ada dua cycle dengan besar 1, sehingga ada dua kemungkinan. Antara mereka memang berasal dari cycle yang besarnya juga 1, atau mereka berasal dari cycle dengan besar 2. Maka, ada dua jawaban, f(0123456789) = **6453781029**, dan f(0123456789) = **6459781023**.

Jawab: **6453781029** atau **6459781023**

12. Ada beberapa observasi yang dapat dilakukan pada soal ini.
    - Perhatikan pernyataan 1, 2, dan 3. Apabila pernyataan 1 bernilai salah, maka pernyataan 2 akan bernilai salah, dan pernyataan 3 akan bernilai benar. Apabila pernyataan 1 bernilai benar, maka pernyataan 2 akan bernilai benar, dan pernyataan 3 akan bernilai salah. Sekarang kita tahu bahwa pernyataan 6, 7, 10, 11, …, 2018, 2019 pasti bernilai benar, karena diatas pernyataan 4 pasti ada pernyataan yang benar dan ada pernyataan yang salah..
    - Pernyataan 2020 pasti salah. Karena pernyataan 2019 adalah benar, maka pernyataan 2017, 2016, 2013, 2012, …, 5, 4, 1 pasti benar.

Sekarang kita dapat menghitung pernyataan apa saja yang salah. Pernyataan yang salah adalah pernyataan 3 dan 2020. Maka, pernyataan yang bernilai benar ada **2018**.

Jawab: 2018

13. Soal ini dapat diselesaikan dengan trik yang cukup menarik. Sebuah angka yang jumlah digitnya habis dibagi 9, pasti habis dibagi 9 (sebagai contoh, 72, 711, 171, dll). Ini dapat dibuktikan dengan induksi. Kelipatan 9 pertama adalah 9, dimana jumlah digit nya sudah 9. Untuk setiap angka X yang ditambah 9 ada beberapa kasus.
    - Apabila digit terakhir X adalah 0, maka digit terakhir akan menjadi 9.
    - Apabila digit terakhir X bukan 0, maka digit ini akan berkurang 1, dan digit kedua terakhir akan bertambah 1. Kasus ini dapat menjadi dua kasus lagi
        - Apabila digit kedua terakhir bukan 9, maka nilai digit ini akan bertambah 1.
        - Apabila digit kedua terakhir adalah 9, digit ini akan menjadi 0, dan digit selanjutnya akan bertambah 1. Perhatikan bahwa kasus ini akan berulang secara rekursif.

    Dari semua kasus yang ada, jumlah digit X modulo 9 tidak berubah. Maka dari itu, angka 9 dan kelipatannya pasti memiliki jumlah digit yang habis dibagi 9.

    Perhatikan fakta bahwa suatu angka X, jumlah digit nya tidak akan berubah apabila ditambahkan 9. Kita dapat menggunakan fakta ini untuk mencari kesimpulan bahwa untuk semua angka X, nilai X modulo 9 sama dengan nilai jumlah digit X modulo 9. Ini dikarenakan kita bisa menggunakan angka awal apapun dari 1 sampai 8 (angka tersebut jika di-modulo 9 tidak berubah), dan jika kita tambahkan 9 berapa kali pun, jumlah digit nya modulo 9 juga tidak berubah.

    Angka 2^{29} memiliki 9 digit yang berbeda semua. Maka, hanya ada satu digit yang tidak ada di angka tersebut. Pertama-tama kita cari tau 2^{29} modulo 9. Kita dapat menggunakan prinsip modulo dimana (a * b) modulo m = ((a modulo m) * (b modulo m)) modulo m.
    - $2^{29} = (2^{14} * 2^{14}) * 2$
    - $2^{14} = (2^7 * 2^7)$
    - $2^7 = (2^3 * 2^3) * 2 = (8 * 8) * 2 = 64 * 2 = 128$. 128 modulo 9 = 2
    - $2^{14} = (2 * 2) = 4$. 4 modulo 9 = 4.
    - $2^{29} = (4 * 4) * 2 = 16 * 2 = 32$. 32 modulo 9 = 5

Dengan hasil tersebut, kita dapat menyimpulkan bahwa jumlah digit 2^{29} modulo 9 adalah 5. Satu-satu nya angka 9 digit yang memenuhi syarat yang telah diberikan adalah jika angka 9 digit tersebut tidak memiliki digit 4 (jumlah digit nya adalah 41, dan 41 modulo 9 = 5).

Jawab: 4

14. Terdapat 393 bilangan dari 1 hingga 1000 yang merupakan kelipatan 3 dan kelipatan x. Kita dapat mengetahui berapa banyak bilangan yang merupakan kelipatan 3. Mencari banyaknya bilangan kelipatan 3 dari 1 hingga 1000 adalah 1000/3 = 333(dibulatkan kebawah). Dengan menggunakan cara yang sama, kita juga bisa mencari banyaknya bilangan kelipatan x yaitu 1000/x. Dari angka 1 hingga 1000, kita bisa melihat bahwa ada bilangan 3x disana(karena 3x merupakan kelipatan 3 dan juga merupakan kelipatan x). Artinya semua bilangan kelipatan 3x akan terhitung dua kali di kelipatan 3 dan kelipatan x. Maka dari itu, kita harus mengurangi sebanyak kelipatan 3x agar tidak terjadi penghitungan dua kali. Dari hal tersebut, maka kita akan mendapatkan
1000 div 3 + 1000 div x - 1000 div (3x) = 393
333 + 1000 div x - 1000 div (3x) = 393
1000 div x - 1000 div (3x) = 60
x = 11

Jawab: 11

15. Kita dapat menyelesaikan soal ini dengan simulasi. Observasi awal yang dapat kita lihat adalah kita dapat melihat semua kota yang bisa dikunjungi saat dilakukan tepat i penerbangan dan dikunjungi dari berapa rute. Simulasi sebagai berikut,

tepat 0 penerbangan:

0 tetap berada di 0 => 1 cara

tepat 1 penerbangan:

0 pergi ke 1 => 1 cara
0 pergi ke 2 => 1 cara

0 pergi ke 6 => 1 cara
maka untuk tepat 1 penerbangan, ada 1 cara ke 1, 1 cara ke 2, dan 1 cara ke 6.

tepat 2 penerbangan:

1 pergi ke 3 => 1 cara
1 pergi ke 2 => 1 cara
2 pergi ke 3 => 1 cara
2 pergi ke 5 => 1 cara
2 pergi ke 7 => 1 cara
6 pergi ke 2 => 1 cara
6 pergi ke 7 => 1 cara
maka untuk tepat 2 penerbangan, ada 1+1 cara ke 3, 1+1 cara ke 2, 1 cara ke 5, 1+1 cara ke 7.

tepat 3 penerbangan:

2 pergi ke 3 => 2 cara
2 pergi ke 5 => 2 cara
2 pergi ke 7 => 2 cara
3 pergi ke 4 => 2 cara
3 pergi ke 8 => 2 cara
5 pergi ke 3 => 1 cara
5 pergi ke 4 => 1 cara
5 pergi ke 8 => 1 cara
5 pergi ke 7 => 1 cara
7 pergi ke 8 => 2 cara
7 pergi ke 4 => 2 cara
maka untuk tepat 3 penerbangan, ada 2+1 cara ke 3, 2+1+2 cara ke 4, 2 cara ke 5, 2+1 cara ke 7, 2+1+2 cara ke 8.

tepat 4 penerbangan:

3 pergi ke 4 => 3 cara

3 pergi ke 8 => 3 cara

4 pergi ke 9 => 5 cara

5 pergi ke 4 => 2 cara

5 pergi ke 9 => 2 cara

5 pergi ke 8 => 2 cara

7 pergi ke 4 => 3 cara

7 pergi ke 8 => 3 cara

8 pergi ke 9 => 5 cara

maka untuk tepat 4 penerbangan, ada 3+2+3 cara ke 4, 3+2+3 cara ke 8, 5+2+5 cara ke 9.
Maka, untuk tepat 4 penerbangan yang ke 9, ada **12** cara.

Jawab: 12

16. Soal tersebut dapat kita artikan sebagai "Berapa banyak rute penerbangan maksimal yang harus dipasang agar terdapat tepat 2 rute dari 0 ke 9". Untuk masalah tersebut, kita bisa mencari rute terpanjang dari 0 ke 9. Untuk mencari rute terpanjang, kita dapat menggunakan dynamic programming. Untuk suatu node, rute terpanjang dapat dihitung dengan :
dp[kota] = max(dp[pergi_ke1],dp[pergi_ke2],dp[pergi_ke3],...).
Dengan cara tersebut, kita bisa melihat arah dari rutenya menjadi 0->1->2->3->4->9.
Jika diperhatikan dengan seksama, graph yang diberi memiliki kesimetrisan. Sehingga rute lainnya yang merupakan rute terpanjang adalah 0->6->2->7->8->9.
Sehingga, banyaknya rute yang harus dibuang adalah semua rute yang tidak termasuk dari 2 rute tersebut, yaitu 11.

Jawab: 11

17. Peluang untuk angka pada suatu posisi sama adalah 1/2. Maka peluang untuk angka sepanjang 8 angka adalah $(1/2)^8$. Maka nilai n adalah **8**.

Jawab: 8

18. Dengan menggunakan algoritma dijkstra, maka akan didapat sebesar 15. Untuk mengetahui apa itu dijkstra, silahkan melihat ke link berikut. (karena soal klasik, langsung refer ke soal yg mirip(?))

19. Untuk mengetahui pertanyaan minimum yang harus dikerjakan sebelum mengerjakan pertanyaan x, kita harus mencari berapa pertanyaan minimum yang harus dikerjakan sebelum mengerjakan pertanyaan dari setiap prasyarat pertanyaan x. Kita dapat melakukannya dengan mencoba mencari dari pertanyaan yang bisa langsung dikerjakan, yaitu 10. saat pertanyaan 10 dikerjakan, maka kita dapat mengerjakan soal nomor 5 atau 7 atau 8. Saat kita coba memilih salah satu dari pertanyaan nomor 5, 7, dan 8, kita dapat memiliki pilihan untuk mengerjakan pertanyaan nomor 2, 3, 4, dan 9. Saat kita coba memilih salah satu pertanyaan dari 2,3,4, dan 9, maka kita dapat memiliki pilihan untuk mengerjakan pertanyaan nomor 1 dan 6. Target kita adalah 6, sehingga kita harus menjawab setidaknya **3** pertanyaan sebelum menjawab pertanyaan nomor 1.

Jawab: 3

20. Karena A bernilai TRUE, maka B bernilai TRUE(premis 1). Karena A bernilai TRUE, maka D bernilai TRUE(premis 4). Karena D bernilai TRUE, maka C bernilai FALSE (premis 3). Karena B bernilai TRUE, maka premis 2 tidak memberi kontribusi apa-apa. Maka hanya ada 1 kemungkinan nilai B, C, dan D sehingga kelima premis tersebut benar.

Jawab: 1

21. Untuk mengerjakan soal ini, kita dapat memanfaatkan solusi dari nomor 20. Pada soal nomor 20, kita tidak memperhitungkan nilai E. Jika kita perhitungkan nilai E, maka akan ada 2 kemungkinan jika A bernilai TRUE (premis 5 tidak memberi pengaruh karena nilai D adalah TRUE). Jika A bernilai FALSE, maka nilai D akan menjadi FALSE (premis 4). Karena D bernilai FALSE, maka E akan bernilai FALSE(premis 5). Premis 1 tidak akan memberi dampak karena A bernilai FALSE. premis 3 juga tidak akan memberi pengaruh karena D bernilai FALSE. Maka tersisa premis 2 dimana jika B bernilai FALSE, maka C akan menjadi FALSE. Tetapi jika B bernilai TRUE, maka nilai C bisa menjadi TRUE ataupun FALSE. Maka terdapat 3 kemungkinan jika A bernilai FALSE. Total kombinasi yang berbeda akan menjadi 5 kombinasi.

Jawab: 5

22. Konfigurasi untuk membuat langkah terkecil adalah disaat konfigurasi kartu terurut menaik, yang artinya tidak ada perubahan yang perlu dilakukan (0 langkah).

Jawab: 0

23. Konfigurasi terburuk dapat terjadi jika susunan awal terurut menurun.
Jika kita hanya memiliki 1 kotak, maka kita tidak perlu memindahkan kotak.
Jika kita hanya memiliki 2 kotak, maka kita bisa menukar angka 2 ke ujung kanan dengan 1 cara.
Jika kita hanya memiliki 3 kotak, maka kita bisa menukar angka 3 ke ujung kanan dengan 1 cara dan menyelesaikan hal yang sama seperti 2 kotak. Maka total langkah adalah 3 cara.
Jika kita hanya memiliki 4 kotak, maka kita bisa menukar angka 4 ke ujung kanan dengan 3 cara dan menyelesaikan hal yang sama seperti 3 kotak. Maka total langkah adalah 6 cara.
Maka jika kita memiliki 10 kotak, maka total langkah yang kita lakukan adalah 1+2+3+4+5+6+7+8+9 = 45 langkah.

Jawab: 45

24. Kita dapat melakukan simulasi untuk mendapatkan A[0] hingga A[11]. Sehingga dapat dihitung menjadi 0+0+1+3+0+4+1+7+0+8+1+11 = 36.

Jawab: 36

25. Untuk nomor 25, kita tidak bisa melakukan cara yang sama dengan nomor 24 karena akan terlalu lama. Kita harus mengetahui cara mencari A[0] dengan cepat. Untuk mencari XOR dari 1 hingga N, terdapat pola dimana jika N modulo 4 = 0 maka akan bernilai N, jika N modulo 4 = 1 maka akan bernilai 1, jika N modulo 4 = 2 maka akan bernilai N+1, dan jika N modulo 4 = 3 maka akan bernilai 0. A[0] merupakan XOR dari 1 hingga (N-1), maka A[0] akan bernilai 0. A[1] merupakan A[0] yang di XOR dengan 0, A[2] merupakan A[1] yang di XOR dengan 1, dan berlaku seterusnya. Jika f(x) adalah XOR dari 1 hingga x, maka kita harus mencari nilai dari 0 + 0 XOR f(0) + 0 XOR f(1) + ... + 0 XOR f(N-2). Karena 0 XOR x = x, maka kita dapat mengabaikan 0. Untuk mempermudah penghitungan, kita dapat mengelompokan menjadi (f(0) +

f(1) + f(2) + f(3)) + ... + (f(192) + f(193) + f(194) + f(195)) + (f(196) + f(197) + f(198)). Untuk setiap (f(4*i) + f(4*i+1) + f(4*i+2) + f(4*i+3)) = (4*i + 1 + 4*i+2+1 + 0) = (8*i+4). Untuk mencari jumlah dari f(0) hingga f(195), kita dapat menggunakan rumus deret aritmatika dengan U1 = 4, b(Ui - U(i-1)) = 8 dan terdiri dari 49 bilangan. Maka akan didapat (U1 + Un)*n/2 = (4 + 4 + 8*48)*49/2 = 9604. Lalu kita akan mencari f(196) + f(197) + f(198) secara manual, yaitu 196+1+199 = 396. Jadi totalnya adalah 9604 + 396 = 10000.

N.B: untuk N berkelipatan 4, $F(N) = (N/2)^2$. Pembuktian diserahkan kepada pembaca sebagai latihan.

Jawab: 10000

26. Disaat bebek ke X mengambil ikan, maka sudah ada sebanyak 1 + 3 + 5 + ... + 2*X-1 ikan yang diambil. Kita dapat mengetahui bahwa ini adalah penjumlahan X bilangan ganjil pertama yang ekuivalen dengan X^2 (deret aritmatika). Maka dari itu, kita bisa tahu bahwa X^2 + 225 = N^2 untuk N dan X bilangan bulat. Kita pecah persamaan tersebut sebagai berikut:
N^2 - X^2 = 225
(N+X)(N-X) = 225
Maka N+X dan N-X merupakan faktor dari 225. 225 memiliki 9 pasang faktor. Karena nilai N dan X merupakan bilangan bulat lebih besar dari 0, maka nilai (N+X)+(N-X) harus genap. Karena semua faktor dari 225 adalah ganjil, maka jumlah dari setiap pasang faktor pasti adalah genap dan akan menjamin bahwa N dan X adalah bilangan bulat. Kita juga harus menjamin bahwa (N+X) harus lebih besar dari (N-X) karena nilai X tidak bisa negatif. Jadi dari 9 pasang faktor, hanya ada 4 faktor yang memenuhi kondisi tersebut, yaitu (1, 225), (3, 75), (5, 45), dan (9, 25).
Dari setiap pasang faktor, kita dapat mencari nilai N dengan cara menjumlahkan kedua faktor tersebut. (N-X) + (N+X) = 2*N. Sisanya dapat diselesaikan dengan aljabar, sehingga kita dapat mengetahui nilai X untuk setiap pasang faktor.
    - 2*N = 1 + 225. N = 113. N-X = 1. X = 112.
    - 2*N = 3 + 75. N = 39. N-X = 3. X = 36.
    - 2*N = 5 + 45. N = 25. N-X = 5. X = 20.
    - 2*N = 9 + 25. N = 17. N-X = 9. X = 8.

Maka, hasil penjumlahan dari semua nilai X yang mungkin adalah **176**.

Jawab: 176

27. Soal ini apabila dikerjakan dari yang diminta (10 bebek) akan membutuhkan waktu yang sangat lama, karena kasus yang mungkin terjadi sangat banyak. Akan tetapi, apabila kita mengerjakan dari bawah, maka kita bisa mengerjakan dengan lebih mudah. Metode ini memiliki nama, yaitu Bottom-Up. Sedangkan metode yang tadi (mengerjakan soal dari yang diminta) adalah Top-Down.

Apabila tersisa 1 bebek saja, maka 10! ikan akan langsung ia ambil. Apabila tersisa 2 bebek, maka bebek termuda akan menolak tawaran tersebut. Hal ini dikarenakan pada saat tersisa 2 bebek, bebek termuda hanya mendapatkan 10!/2 ikan. Dengan menolak tawaran tersebut, ia memiliki kemungkinan untuk mendapatkan jumlah ikan yang lebih banyak. Akan tetapi, bebek yang lebih tua tidak ingin mendapatkan 0 ikan. Oleh karena itu, ia harus menerima tawaran tersebut. Karena terdapat 50% bebek yang setuju, maka permainan berakhir, dan masing-masing bebek mendapatkan 10!/2.

Apabila tersisa 3 bebek saja, maka bebek termuda dan kedua termuda akan menolak tawaran tersebut. Pada putaran tersebut, mereka hanya mendapatkan 10!/3 ikan saja, sedangkan jika mereka menolak, mereka bisa mendapatkan 10!/2. Oleh karena itu bebek termuda dan kedua termuda menolak tawaran tersebut, sehingga bebek ketiga termuda tidak mendapatkan ikan apapun.

Apabila tersisa 4 bebek saja, maka bebek termuda dan kedua termuda akan menolak tawaran tersebut, karena mereka hanya mendaptkan 10!/4. Apabila mereka berhasil melanjutkan permainan ke 3 bebek, maka mereka akan mendapatkan 10!/2. Akan tetapi, jika bebek ketiga termuda menolak, maka ia tidak akan mendapatkan ikan. Oleh karena itu, bebek ketiga termuda harus menerima tawaran tersebut untuk mendapatkan kemungkinan menerima 10!/4. Demikian pula dengan bebek keempat termuda. Karena terdapat 50% bebek yang setuju, maka permainan berakhir, dan masing-masing bebek mndapatkan 10!/4.

Anda bisa melanjutkan proses tersebut hingga ke jumlah bebek yang lebih banyak. Apabila tersisa 5, 6, dan 7 bebek, maka mereka tidak dapat mendapatkan ikan apapun. Karena bebek termuda hingga termuda keempat bisa mendapatkan 10!/4, tetapi jika mereka menerima tawaran itu, jumlah ikan yang mereka dapatkan akan berkurang. Oleh karena itu, mereka akan selalu menolak tawaran itu. Kemudian apabila tersisa 8 bebek, maka bebek termuda ke-5, 6, dan 7 harus menerima tawaran itu. Karena kalau tidak, mereka tidak akan mendapatkan ikan. Karena terdapat 50% bebek yang setuju, maka permainan berakhir, dan masing-masing bebek mendapatkan 10!/8.

Apabila digeneralisasikan, maka pada saat tersisa 2^n bebek, 2^{n-1} bebek yang ada akan setuju, dan sisanya akan menolak. Karena terdapat 50% bebek yang setuju, maka permainan berakhir. Dengan kata lain, permainan akan selalu berakhir apabila jumlah bebek yang ada adalah dalam bentuk 2^n. Tanpa generalisasi ini pun, Anda tetap bisa mengerjakan soal nomor ini, karena jumlah bebek awal yang ditanyakan hanya 10.
Karena itu, nilai 2^n terbesar yang lebih kecil atau sama dengan 10 adalah **8.**

Jawab: 8

28. Pada dasarnya, soal ini merupakan soal *Shortest Path*. Masing-masing poin yang tertera pada kotak merupakan biaya (dalam soal ini, jumlah ayam yang disapa) yang dibutuhkan. Kita dapat memperoleh jawabannya dengan mencari *shortest path* dari A5 ke E1 menggunakan biaya yang tersedia pada masing-masing kandang. Apabila ladang tersebut direpresentasikan dalam sebuah *graph*, maka *graph* tersebut adalah *weighted graph*, yang mana weight nya adalah jumlah ayam yang disapa. Untuk menyelesaikan *shortest path* pada *weighted graph*, Anda dapat menggunakan algoritma Dijkstra.

Kemudian, karena yang diminta oleh soal adalah lintasan yang Kwik lalui, bukan hanya jumlah menyapa ayam minimum. Untuk memperoleh lintasan tersebut, kita dapat melakukan Backtracking. Setelah kita melakukan algoritma Dijkstra, kita sudah memiliki data jalur terpendek dari A5 ke semua kandang. Apabila kita mulai mencari lintasan dari A5, kita tidak tahu jalur mana yang dipakai untuk menuju ke E1. Tetapi apabila kita mulai mencari lintasan dari E1 dan bergerak mundur, kita tahu bahwa jalur yang dipakai adalah kandang yang dikunjungi sebelum mengunjungi kandang E1. Lakukan tahapan mundur ini hingga Anda mencapai A5, dan

itulah lintasan yang diambil. Karena bergerak mundur ini, algoritma ini diberi nama Backtracking.

Jawab: A5-B5-B4-C4-C3-D3-D2-D1-E1.

29. Untuk mengerjakan soal ini, kita dapat mencari harga minimum apabila menggunakan masing-masing angkot. Misalkan kita menaiki angkot biru. Karena angkot berangkat setiap 5 menit, Pak Dengklek dapat menaiki angkot pada jam 8.10 dan tiba di pasar pada jam 8.50. Harga pakan pada jam tersebut adalah 140 ribu Rupiah. Apabila ditambah dengan harga angkot, yaitu 5 ribu Rupiah, maka pengeluaran total Pak Dengklek adalah 145 ribu Rupiah apabila Pak Dengklek menaiki angkot biru. Lakukan cara yang sama untuk ketiga angkot yang lain, kemudian ambil harga yang paling minimum.

Jawab: Merah.

30. Soal ini adalah tipe soal simulasi. Dengan kata lain, kita dapat menyelesaikan soal ini hanya dengan mengikuti apa yang diminta oleh soal. Pada nama “Hilbert”, huruf pertama nama tetap dan kapital, yaitu H. Kemudian huruf “e” dan “i” dapat dihapus, menjadi “Hlbrt”. Huruf “l” diganti dengan 4, “b” diganti dengan 1, “r” diganti menjadi 6, dan “t” diganti menjadi 3.

Jawab: H4163.

31. Kita ketahui bahwa kunci kombinasi tersebut diawali dengan 110 dan diakhiri oleh 10. Dengan kata lain, terdapat 5 digit yang kita belum ketahui nilai nya. Berdasarkan informasi pertama, terdapat 2 deretan 010 secara terpisah, yang mana keduanya secara total membutuhkan 6 posisi. Karena hanya terdapat 5 posisi yang tersisa, kita ketahui bahwa deretan 010 ini pasti bertumpukan dengan awalan 110 atau akhiran 10.

Apabila kita menumpuk awalan 110 dengan deretan 010, maka awalan dari kunci kombinasi tersebut menjadi 11010. Konfigurasi kunci kombinasi dengan awalan 11010 dan akhiran 10 adalah 11010XXX10. Beberapa kunci kombinasi yang mungkin adalah 1101001010 atau 1101010010. Terdapat beberapa jawaban yang benar untuk soal ini.

Jawab: 1-1-0-1-0-0-1-0-1-0 atau **1-1-0-1-0-1-0-0-1-0** atau **1-1-0-0-1-0-1-0-1-0**

32. Untuk menarik kesimpulan, terdapat beberapa observasi yang dapat kita perhatikan. Pertama, dengan melihat informasi kedua dan keempat, kita dapat memutuskan bahwa 5 bukanlah angka yang benar. Hal ini dikarenakan apabila 5 benar, maka posisi nya berada di posisi kedua, namun hal ini kontradiksi dengan pernyataan "posisi salah" pada informasi kedua. Karena 5 bukanlah angka yang benar, maka 8 dan 4 adalah angka yang benar.

    Kemudian, kita dapat membandingkan informasi kedua dan ketiga. Kita ketahui bahwa 8 adalah salah satu digit yang benar. Berdasarkan informasi kedua, angka 8 tidak berada di posisi pertama. Demikian pula berdasarkan informasi ketiga, angka 8 tidak berada di posisi kedua. Dengan kata lain, angka 8 berada di posisi ketiga.

    Terakhir, kita dapat membandingkan informasi pertama dan keempat. Kita ketahui bahwa 8 dan 4 adalah digit yang benar. Dengan kata lain, hanya tersisa sebuah angka yang belum kita ketahui, dan digit tersebut harus berada di informasi pertama dan keempat. Angka tersebut adalah 7. Berdasarkan informasi keempat, dapat kita simpulkan 7 berada di posisi pertama. Sisanya, angka 4 berada di posisi kedua.

    Jawab: 7-4-8.

33. Untuk menyelesaikan soal ini, kita dapat melakukan *bruteforce*. Akan tetapi, apabila kita melakukan bruteforce total, banyaknya konfigurasi yang perlu kita periksa akan sangat banyak, sehingga kita membutuhkan cara *bruteforce* yang lebih baik. Observasi pertama yang dapat membantu *bruteforce* kita adalah: perhatikan bahwa apabila desa lokasi akhir Kwik adalah C, maka Kwik pasti dapat berada di A. Observasi ini akan mengurangi jumlah *bruteforce* yang kita lakukan, dengan tidak perlu memperhatikan apabila jawabannya adalah A. Dengan cara yang sama, apabila desa lokasi akhir Kwik adalah E, maka Kwik pasti dapat berada di F.

    Jawab: F

34. Bayangkan apabila Anda berada pada *maze* tersebut dan mengikuti peraturan yang diberikan. Seakan-akan, Anda selalu menyentuh dinding yang ada di sisi kanan Anda. Apabila Anda terus berjalan mengikuti aturan tersebut, maka semua bagian dinding yang terhubung dengan dinding awal Anda akan dapat Anda kunjungi. Sebagai contoh, dijamin bahwa pada *maze* C, Anda tidak akan bisa mencapai pintu keluar, karena tembok yang Anda sentuh tidak terhubung dengan tembok di pintu keluar. Tentu saja selain observasi ini, Anda juga harus memeriksa apakah pintu keluar memang dapat dicapai meskipun tanpa menggunakan aturan yang diberikan.

Jawab: C.

35. Untuk menyelesaikan soal ini, kita dapat melakukan *bruteforce*. Akan tetapi, apabila kita melakukan *bruteforce* total, banyaknya konfigurasi yang perlu kita periksa akan sangat banyak, sehingga kita membutuhkan cara *bruteforce* yang lebih baik. Salah satu observasi penting dari soal ini adalah, 5 biskuit pertama yang Kwek dapatkan adalah P, Q, S, R, T, yang mana biskuit-biskuit tersebut adalah 1 dari masing-masing kandang yang ada. Observasi ini mungkin tidak memberikan jawaban langsung terhadap soal, tetapi akan mempermudah proses *bruteforce* Anda.

Pertama, apabila biskuit Q disediakan di kandang kambing, maka dijamin akan selalu ada jalan antar kandang ayam/antar kandang kelinci yang kita lewati. Diketahui pada soal bahwa jalan tersebut membutuhkan waktu yang paling lama, yaitu 7 menit. Untuk mencari waktu paling minimum, kita usahakan untuk mengabaikan jalan ini apabila mungkin.

Apabila biskuit S disediakan di kandang kambing, terdapat 1 konfigurasi yang tidak menggunakan jalan antar kandang ayam/antar kandang kelinci, yaitu: P untuk ayam kiri bawah, Q untuk kelinci kanan bawah, S untuk kandang kambing, R untuk kelinci kiri atas, T untuk ayam kanan atas. Waktu total yang dibutuhkan untuk jalur ini adalah 14 menit.

Apabila biskuit R disediakan di kandang kambing, terdapat beberapa konfigurasi yang tidak menggunakan jalan antar kandang ayam/antar kandang kelinci. Akan tetapi, karena jalan dari R ke T digunakan 4 kali (SR, RT, TR, RP), waktu yang dibutuhkan hanya untuk ini saja adalah 12 menit. Konfigurasi-konfigurasi pada kasus ini akan tidak terlalu efisien.

Jawab: S

36. Perhatikan bahwa fungsi printgraf(x) akan memanggil fungsi printgraf(x-1) sebanyak 2 kali, dan fungsi printgraf(x-2) sebanyak 1 kali, kecuali jika x bernilai negatif. Kita definisikan sebuah fungsi f(x) yang mengembalikan nilai dari banyaknya pemanggilan fungsi printgraf() apabila dipanggil printgraf(x-1). Dengan kata lain, secara rekursif dapat kita definisikan f(x)=2f(x-1)+f(x-2) untuk x>=0, f(x)=1 untuk x<0. Karena dipanggil printgraf(6), maka jawaban yang kita inginkan adalah f(6).

Jawab: 865

37. Ide untuk menyelesaikan soal ini mirip dengan soal nomor 36. Kita definisikan sebuah fungsi g(x) yang mengembalikan nilai dari banyaknya karakter bintang yang dicetak apabila dipanggil fungsi printgraf(x). Secara rekursif, dapat kita definisikan g(x)=2g(x-1)+g(x-2)+x untuk x>=0, f(x)=0 untuk x<0. Jawaban yang kita inginkan adalah nilai x terkecil sehingga g(x)>=2019. Karena barisan yang dihasilkan fungsi g(x) berkembang secara eksponensial, nilai dari x tidak akan terlalu besar untuk mencapai g(x)>=2019. Sehingga kita dapat melakukan *bruteforce*, hingga mendapatkan nilai x pertama yang mana g(x)>=2019.

Jawab: 9

38. Perhatikan bahwa perulangan dengan menggunakan variabel i menandakan perulangan pada masing-masing baris pada output, yaitu dari 1 hingga N. Hal ini dikarenakan pada terdapat writeln pada akhir perulangan yang membuat output tersebut ganti baris. Selanjutnya, kita dapat menganalisa apa yang dilakukan oleh perulangan-perulangan pada variabel j.

    Pada program A, untuk baris ke-i, terdapat i buah karakter spasi dan N-i+1 buah karakter bintang. Pada program B, untuk baris ke-i, terdapat N-i+1 buah karakter spasi dan i buah karakter bintang. Pada program C, untuk baris ke-i, terdapat i-1 buah karakter spasi dan N-i+1 buah karakter bintang. Pada program D, untuk baris ke-i, terdapat i buah karakter spasi dan N-i buah karakter bintang. Perhatikan bahwa pada program A, terdapat kelebihan 1 karakter spasi di awal, sehingga output dari program A tidak sesuai dengan yang diminta.

Jawab: C

39. Untuk mengerjakan soal ini dibutuhkan pemahaman akan cara komputer menjalankan fungsi rekursif. Pada pemanggilan fungsi whats(x), g akan bernilai 2x+1, sedangkan h akan bernilai 2x. Dengan kata lain, apabila kita memanggil fungsi whats(x), maka program akan memanggil fungsi what(2x+1), kemudian mencetak nilai dari x, lalu memanggil fungsi what(2x). Jenis rekursi ini juga dikenal dengan nama *Infix*.

    Sebagai contoh, apabila n=3, pada saat kita memanggil whats(1), program akan menjalankan whats(3), lalu mencetak angka 1, lalu mejalankan whats(2). Pada saat program memanggil fungsi whats(3), program akan memanggil whats(7), lalu mencetak angka 3, lalu memanggil whats(6). Perhatikan bahwa tidak terjadi apapun pada pemanggilan whats(7), karena tidak memenuhi syarat x<=n. Karena pemanggilan whats(7) sudah selesai maka program mencetak angka 3. Inilah angka pertama yang dicetak. Kemudian tidak terjadi apapun pada pemanggilan fungsi whats(6), karena tidak memenuhi syarat x<=n. Dengan ini, pemanggilan fungsi whats(3) selesai. Setelah fungsi whats(3) selesai, angka 1 akan dicetak (dari pemanggilan whats(1)). Selanjutnya, whats(2) akan dipangil. Mirip dengan pemanggilan whats(3), pada pemanggilan whats(2), tidak terjadi apa-apa pada pemanggilan whats(5) dan whats(4). Akhirnya, urutan angka yang dicetak adalah 3, 1, dan 2.

    Dengan cara yang sama, Anda bisa melakukan *bruteforce* untuk n=10. Untuk mempermudah pengerjaan, Anda dapat menggambar *Recursion Tree* sebagai visualisasi. Dua angka pertama yang dicetak adalah 7 dan 3, sedangkan dua angka terakhir yang dicetak adalah 4 dan 8.

    Jawab: 7,3,4,8.

40. Untuk mengerjakan soal nomor 40, bruteforce akan memakan waktu yang sangat banyak apabila Anda menjalankan semua rekursi yang ada. Perhatikan bahwa whats(2x+1) akan terus dipanggil lebih awal, sedangkan whats(2x) akan dipanggil lebih akhir. Fungsi whats(1) akan memanggil whats(3), whats(3) akan memanggil whats(7), whats(7) akan memanggil whats(15), …, whats(127) akan memanggil whats(255). Anda juga dapat melihat bahwa fungsi yang dipanggil adalah x dalam bentuk 2^n-1. Dengan kata lain, bilangan pertama yang dicetak adalah 255. Bilangan kedua yang dicetak adalah 127 (karena whats(127) yang memanggil whats(255)).

Untuk 2 bilangan terakhir, dapat Anda lakukan cara yang sama, tetapi whats(2x) alih-alih dari whats(2x+1). Fungsi whats(1) akan memanggil whats(2), whats(2) akan memanggil whats(4), whats(4) akan memanggil whats(8), …, whats(128) akan memanggil whats(256). Fungsi yang dipanggil adalah x dalam bentuk 2^n. Dengan kata lain, bilangan terakhir yang dicetak adalah 256, dan bilangan yang dicetak kedua terakhir adalah 128.

Untuk bilangan ketiga, keempat, ketiga terakhir, dan keempat terakhir dapat dikerjakan dengan cara yang sama. Untuk bilangan ketiga: setelah 127 dicetak, whats(127) akan memanggil whats(254). Inilah bilangan ketiga yang dicetak. Setelah whats(127) selesai, maka program akan kembali ke fungsi yang memanggil whats(127), yaitu whats(63), kemudian mencetak bilangan 63. Inilah bilangan keempat yang dicetak. Untuk ketiga terakhir dan keempat terakhir diserahkan kepada pembaca sebagai latihan.

Jawab: **255,127,254,63,253,258,64,257,128,256**.

Bahasa yang digunakan untuk soal bagian C adalah C++

1. Kita dapat menghitung semua bilangan fibonacci dari fib(0) hingga fib(100000) dengan cuplikan kode sederhana ini.

```
// array yang akan digunakan untuk menyimpan bilangan
fibonacci
int fib[1000005];

// di dalam program utama, atau fungsi pembantu
fib[0] = 0;
fib[1] = 1;
for(int i=2; i<=100000; i++)
    fib[i] = fib[i-1] + fib[i-2];
```

Karena di soal diminta untuk hanya mengeluarkan hasil nya modulo $10^9 + 7$, kita dapat menggunakan prinsip modulo dimana (a + b) modulo m = ((a modulo m) + (b modulo m)) modulo m, untuk menghindari integer overflow. Ini berarti, sebelum menyimpan nilai fibonacci ke-i ke dalam array, kita bisa melakukan operasi modulo terlebih dahulu.

```
// deklarasi MOD di luar program utama
const int MOD = 1e9 + 7;

for(int i=2; i<=100000; i++)
    fib[i] = (fib[i-1] + fib[i-2]) % MOD;
```

Dengan cuplikan ini, kita dapat menjawab semua pertanyaan hanya dengan melihat array yang sudah dibuat ini.

```
int main(){
    // generate array fib[]

    int q;
    cin >> q;
    for(int i=1; i<=q; i++){
        int n;
        cin >> n;
        cout << fib[n] << endl; // lihat array
    }
    return 0;
}
```

Analisis kompleksitas: Untuk mencari semua bilangan fibonacci, kompleksitasnya adalah O(N). Ini cukup mudah analisis nya karena program untuk mencari semua bilangan fibonacci hanya menggunakan satu pengulangan dasar. Untuk menjawab sebuah pertanyaan, kompleksitas nya adalah O(1), karena hanya melakukan akses array. Maka, kompleksitas akhirnya adalah O(N + Q).

2. Kita dapat membuat semua kemungkinan garis yang ada dan mengelompokkannya sesuai dengan gradien mereka. Berikut adalah program yang mencari semua pasangan titik yang mungkin, beserta dengan input program.

```
int n;
int x[2500], y[2500];

void buat_garis(){
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            if(i == j) continue;
            tambah_garis(i, j); // fungsi yang akan datang
        }
    }
}

int main(){
    cin >> n;
    for(int i=0; i<n; i++){
        cin >> x[i] >> y[i];
    }
```

```
    buat_garis();

    return 0;
}
```

Untuk melakukan pengecekan apakah gradien suatu garis dengan garis lainnya sama, kita dapat menggunakan tipe data double. Ini karena tipe data double dapat menyimpan hingga 15 angka dibelakang koma, dan sola hanya meminta program ini untuk akurat hingga 9 angka di belakang koma.

```
int m = 0; // banyak garis, awalnya ada 0 garis
double gradien[2500 * 2500];

void tambah_garis(int a, int b){
    gradien[m++] = (double)(y[a] - y[b]) / (x[a] - x[b]);
}
```

Setelah itu, kita dapat mengurutkan semua garis berdasarkan gradiennya. Setelah pengurutan, garis dengan gradien sama pasti bersebelahan, sehingga untuk mencari semua kelompok yang mungkin, kita hanya perlu melakukan sekali pengulangan.

```
// fungsi pembantu untuk membandingkan dua angka double hingga
// 9 angka di belakang koma
bool sama(double a, double b){
    double beda = b - a;
    if(beda < 0)beda *= -1;
    return beda <= 1e-9;
}

int main(){
    cin >> n;
    for(int i=0; i<n; i++){
        cin >> x[i] >> y[i];
    }

    buat_garis();

    // anda dapat menggunakan algoritma pengurutan favorit
anda
    sort(gradien, gradien + m);

    int awal = 0;
    int ans = 0;
    for(int i=1; i<=m; i++){
        if(i == m || !sama(gradien[awal], gradien[i])){
            // maka garis dengan indeks [awal..i-1] memiliki
```

```
            // gradien sama

            ans = max(ans, i - awal);
            // fungsi max dapat diimplementasikan sendiri
            // atau menggunakan fungsi library c++

            awal = i;
        }
    }

    cout << ans << endl;

    return 0;
}
```

Analisis kompleksitas: Banyak garis yang mungkin adalah sebanyak kurang lebih N*N, sehingga kompleksitas untuk mencari semua garis yang mungkin adalah $O(N^2)$. Kemudian, kompleksitas kebanyakan algoritma sorting adalah O(N log N). Karena besar array yang ingin diurutkan adalah N*N, maka kompleksitas akhirnya adalah $O(N^2 \log N^2) = O(2N^2 \log N) = O(N^2 \log N)$.