Kisi-Kisi dan Materi Uji Olimpiade Sains
BIDANG INFORMATIKA/KOMPUTER

Disertai Contoh-contoh dan Pembahasan

Disusun Oleh:
Tim Pembina Olimpiade Sains
Bidang Komputer

Daftar Isi

I. Pengantar

I.1. Olimpiade Sains Nasional

Dalam beberapa tahun terakhir Departemen Pendidikan Nasional telah meyelenggarakan Olimpiade Sains Nasional (OSN) yang di antaranya terdapat bidang Komputer/Informatika. Pemilihan peserta yang akan bertanding di OSN dilakukan melalui seleksi berjenjang dan serentak di seluruh Indonesia, yaitu:

- tingkat kabupaten/kota (OSK), kemudian
- tingkat provinsi (OSP).

Pada dua tahun terakhir, untuk bidang informatika di tingkat propinsi tercatat diikuti lebih dari 1500 siswa peserta seleksi pertahunnya. Sedangkan di tingkat kabupaten/kota tentunya sekian kali lebih banyak lagi (estimasi kasar ada di atas 8-12 ribuan siswa[1]). Hasil dari seleksi tingkat propinsi menentukan siapa yang akan menjadi salah seorang dari ke 90 siswa peserta OSN.

Selain sebagai ajang prestasi tingkat nasional, OSN bertujuan juga untuk mendapatkan calon peserta pembinaan dan seleksi lebih lanjut hingga dipilih empat siswa terbaik untuk menjadi anggota TOKI (Tim Olimpiade Komputer Indonesia). Mereka itulah yang akan mewakili negara dan bangsa untuk bertanding di tingkat dunia yaitu International Olympiad in Informatics (IOI).

I.2. International Olympiad in Informatics

IOI adalah ajang kompetisi pemrograman di tingkat Internasional yang sudah berlangsung sejak 1985. Indonesia mulai mengikuti IOI sejak 1995. Saat ini IOI diikuti oleh 80-an negara (termasuk semua negara maju) sehingga IOI merupakan lomba paling akbar dalam bidang ini untuk tingkat SMA.

Pada awalnya IOI sendiri adalah lomba murni pemrograman semata berdasarkan masalah-masalah yang sederhana. Para peserta dari berbagai negara secara perseorangan berusaha menyelesaikan sejumlah masalah dalam waktu yang singkat dengan membuat program penyelesaian masalah. Program yang dihasilkan diuji dengan sejumlah data test (test case) yang mewakili sejumlah kondisi yang mungkin dari input soal tersebut. Program yang dibuat peserta dinilai dari berapa banyak test case yang berhasil dijawab dengan benar oleh program tersebut. Nilai akhir peserta adalah jumlah nilai yang diperoleh dari setiap program yang dibuatnya. Peringkat peserta diurutkan berdasar nilai tersebut dan 1/12 (atau 8.33%) dari

[1] Ini hanya perkiraan kasar saja karena di tingkat kabupaten/kota, penyelenggaraan beserta proses seleksi diserahkan ke masing-masing kabupaten/kota yang bersangkutan sehingga data peserta tidak tercatat dengan lengkap. Sementara, di tingkat propinsi, proses seleksi di lakukan di pusat sehingga bisa diketahui jumlah keseluruhan peserta.

semua peserta pada peringkat teratas peserta mendapatkan medali emas, 1/6 (atau 16.67%) berikutnya mendapatkan medali perak dan 1/3 (atau 33.33%) mendapatkan medali perunggu. Sisanya terdapat 50% yang tidak mendapatkan medali. Peringkat negara disusun atas total perolehan medali peserta.

Dalam perjalanannya problem-problem yang diberikan mengalami peningkatan tingkat kesulitannya terutama sejak akhir tahun 90-an, hingga pada saat ini pemrograman hanya satu aspek kecil semata di dalam lomba ini. Dengan demikian, aspek utama yang diuji adalah kemampuan menyelesaikan masalahnya sendiri. Sehingga bisa dikatakan bahwa kompetisi IOI adalah menguji kemampuan peserta dalam problem solving dengan pemrograman komputer[2]. Setiap peserta dalam waktu yang amat terbatas harus mengerjakan sejumlah masalah yang diberikan dengan menyusun program yang menyelesaikan masalah tersebut.

[2] Harap bagian yang digarisbawahi tersebut dipahami secara lengkap; bukan HANYA menguji kemampuan membuat program komputer, bukan pula HANYA menguji kemampuan menyelesaikan masalah, tetapi KEDUANYA!!!

II. Karakteristik Materi Uji

II.1. Tingkat IOI

Secara umum penyelesaian masalah di tingkat IOI memerlukan aspek-aspek dalam proses berfikir sebagai berikut.

1. Peserta harus mampu membaca deskripsi soal (termasuk input-proses-output) yang dinarasikan sebagai suatu cerita yang di dalamnya terkandung suatu permasalahan yang hendak diselesaikan.
2. Terkait dengan itu, diperlukan juga pemahaman logika yang baik. Agar berdasarkan deskripsi tersebut peserta mampu menyusun model/abstraksi permasalahan. Model dapat berupa interrelasi antar entitas sebagai suatu graf atau bahkan sudah menjadi lebih matang lagi sebagai suatu model atau fungsi rekurens
3. Menemukan metoda dalam penyusunan algoritma berdasarkan model/abstraksi sebelumnya
4. Mampu melakukan optimasi model penyelesaian masalah di aspek 2 dan 3 tersebut untuk mencapai efisiensi algoritma terbaik. Hanya sekedar solusi naïf saja tidak dapat mencapai nilai maksimum)
5. Konversi rancangan algoritma di atas menjadi program serta evaluasi hasil kerja pemrograman di atas berdasarkan seluruh kemungkinan test case yang akan diberikan.
    a. mendeduksi proses dari test case (Input - Output)
    b. mengenali variabilitas test case (kasus ekstrim, kasus sederhana)
6. Melakukan manajemen waktu, memelihara ketelitian dan stamina dalam mengerjakan hal-hal di atas (tahan terhadap presure keterbatasan waktu dan memiliki endurance, keuletan dan ketelitian untuk tidak meloloskan sedikitpun kesalahan)

Sebagai catatan, kemampuan dalam penyusunan program hanyalah salah satu aspek saja, yang lebih sulit adalah dalam kelima aspek-aspek. Efisiensi akan ditentukan dari metodologi apa yang digunakan pada tahap ke 3.

II.2. Tingkat OSK/OSP

Proses seleksi idealnya adalah mengacu model IOI di atas yaitu problem solving dengan pemrograman. Namun, berbeda dengan bidang OSN lain seperti Fisika, Matematika, Kimia dan Biologi, bidang Informatika khususnya pemrograman belum menjadi pelajaran resmi. Kalaupun ada, hanya di sekolah-sekolah tertentu saja dan itupun belum tentu mengajarkan pemrograman. Dalam kurikulum TIK Diknas, pemrograman hanya diberikan dalam porsi yang amat kecil (sebagian besar adalah penggunaan perangkat lunak MS Office).

Oleh sebab itu, materi uji IOI "diterjemahkan" ke dalam materi yang menguji potensi akademis/skolastik tinggi yang relevan dengan aspek-aspek di atas. Diharapkan dari proses seleksi ini, siswa yang berpotensi walaupun belum mahir dalam pemrograman dapat terjaring untuk diberikan pembinaan yang intensif di Pelatnas. Aspek yang sangat bergantung pada ketrampilan peserta dalam pemrograman dikurangi dan digantikan dengan materi uji "analisa dan logika" dan materi uji kemampuan algoritmika.

Tingkatan seleksi OSK-OSP dibedakan atas komposisi dari ketiga komponen materi uji:

- Kemampuan analitika/logika/aritmatika (nonprogramming)
- Kemampuan algoritmika (programming)

Komponen uji pemrograman tidak mungkin untuk diadakan sehingga digantikan dengan kemampuan dan algoritmika saja.[3] Metoda pengujiannya pun tidak bisa dihindari bersifat test obyektif (pilihan ganda). Metoda ini memang banyak sekali kelemahannya yaitu memungkinkan jawaban asal tapi benar, namun, memungkinkan pemeriksaan yang segera dan efisien. Dampak negatif tersebut bisa dikurangi dengan pembuatan soal dan pilihan jawaban yang dirancang dengan matang. Komposisi analitika/logika di tingkat kabupaten/kota adalah yang paling besar.

Di tingkat propinsi pada dasarnya sama dengan di tingkat kabupaten/kota kecuali komposisi algoritmika diperbesar. Ini adalah untuk memacu peserta yang lolos di tingkat kabupaten/kota untuk memperdalam pemahamannya dan ketrampilan prakteknya dalam pemrograman.

II.3. Tingkat OSN

Materi uji semula adalah mirip dengan di tingkat IOI, namun berhubung belum siapnya sebagian besar peserta dalam hal praktek pemrograman maka materi uji pemrograman dikombinasikan dengan materi uji non programming seperti di OSK/OSP.

[3] Uji pemrograman di tingkat provinsi, apalagi di tingkat kabupaten/kota, masih perlu beberapa tahun lagi hingga infrastruktur di setiap daerah sudah merata.

III. Kisi-kisi Materi Nonprogramming

III.1. Umum

Secara umum materi uji tertulis terbagi atas tiga komponen utama: materi uji analitika dan logika, materi uji aritmatika, dan materi uji algoritmika.

- Materi analitika yang bersifat logika bertujuan untuk menguji potensi akademis (skolastik) peserta namun sedapat mungkin memiliki relevansi yang tinggi dengan problem solving dan elemen penting dalam menguasai pemrograman komputer. Kemampuan ini merupakan faktor penting dalam memahami persoalan yang diberikan dan merancang algoritma penyelesaian masalahnya.
- Materi analitika yang bersifat aritmatika sebenarnya sejalan dengan analitika dan logika di atas, karena soal aritmatika disini bukan sekedar menguji ketrampilan dalam hitung-menghitung, tetapi lebih pada cara berpikir yang logis dan analitis namun dengan soal bertemakan aritmatika
- Materi algoritmika bertujuan untuk menguji kemampuan peserta dalam memahami dan menyusun suatu algoritma. Aspek-aspek yang terkait dengan pengetahuan dan bahasa pemrograman direduksi seminimal mungkin ke tingkat pseudocode.

Lebih rinci lagi ketiga kategori tersebut dijabarkan dalam sejumlah aspek sebagai berikut.

III.2. Tipe Soal untuk Menguji Deskripsi Soal

Soal berbentuk cerita untuk menguji kemampuan aspek pertama dan kedua dari proses berfikir IOI pada bagian B.1. (adaptasi dari soal IOI namun untuk dapat diselesaikan secara manual). Soal dibuat untuk mengukur:

- Kemampuan memahami dan mensimulasikan algoritma dalam cerita
- Kemampuan deduksi berdasarkan input menghasilkan output
- Kemampuan deduksi berdasarkan test case (input-output) menghasilkan pemahaman proses.
- Kemampuan menemukan kasus-kasus ekstrim
- Kemampuan optimasi
- Kemampuan menemukan model matematika dari soal

III.3. Tipe Soal Pemahaman Algoritma

Dalam menjawab soal-soal ini peserta harus bisa memahami algoritma yang diberikan dalam notasi pseudopascal dan menelurusi eksekusi algoritma. Kemampuan ini diperlukan sebagai indikator aspek ke-3 dari proses berpikir IOI pada bagian B.1. Jadi soal-soal dibuat untuk mengujur

- Kemampuan memahami konsep elemen konstruksi (if-then-else,loop dan variasinya)
- Kemampuan membaca algoritma secara menyeluruh
- Kemampuan mengeksekusi (termasuk rekursif) dan *process tracing* yang terjadi
- Kemampuan menkonstruksi (coding)

III.4. Tipe soal Kemampuan Dasar Logika

Dalam kaitannya dengan bagian C.1. sejumlah pernyataan logika terkait dengan aspek-aspek sebagai berikut.

- implikasi
- 'jika dan hanya jika'
- kalkulus preposisi
- induksi-deduksi

Pertanyaan yang diajukan dikombinasikan dengan bagian C.1. untuk mengajukan pertanyaan-pertanyaan yang memerlukan penguasaan dasar logika tersebut.

III.5. Menguji kemampuan dasar Aritmatika

Bagaimanapun juga Aritmatika tidak dipisahkan dari problem solving. Namun supaya agak berbeda dari materi uji Olimpiade Matematika maka aspek aritmatika yang akan dipertanyakan adalah yang:

- berisikan unsur langkah-langkah komputasi
- menuntut kemampuan penyusunan model matematis
- keterkaitan dengan sifat dari deret bilangan
- menuntut kemampuan penyusunan model keterkaitan (graf)

III.6. Tipe Soal Kemampuan Dasar Penunjang

Dalam dunia komputasi (problem solving) elemen dasar (matematika) penunjangnya secara teoritis dikenal sebagai matematika diskret[4]. Pada umumnya soal-soal jenis ini akan mengajukan pertanyaan-pertanyaan

[4] Matematika diskret merupakan salah satu cabang ilmu matematika. Soal-soal bertipe C.3 (logika dasar) dan C.4.(aritmatika) sebenarnya termasuk ke dalam bagian ini. Namun berhubung kedua hal

- Himpunan
- Aljabar logika
- Sifat Bilangan (deret)
- Finite State Machine
- Kombinatorik

Pemahaman yang diperlukan bukanlah pemahaman teoritis tetapi pemahaman deduksi atas permasalahan (artinya dengan kemampuan induksi-deduksi maka pertanyaan tetap dapat dijawab tanpa mempelajari matematika diskret secara khusus).

III.7. Lain-lain yang relevan dengan potensi akademis

Terdapat sejumlah soal yang memiliki relevansi dalam menguji potensi akademis yang tidak terkategorikan dalam tipe-tipe soal di atas.

tersebut. memiliki porsi yang relatif cukup besar maka dalam kisi-kisi ini dikelompokkan secara tersendiri-tersendiri.

IV. Penutup

Melalui dokumen ini penulis berharap peserta ataupun pembina peserta dapat menemukan intisari dan tujuan mengenai soal-soal seleksi mulai dari tingkat Kabupaten/Kotamadya, tingkat Provinsi dan tingkat Nasional. Peserta yang lolos ke Tingkat Pelatihan Nasional (Pelatnas) diharapkan merupakan peserta yang memiliki kemampuan analitikal dan merancang algoritma yang tinggi. Referensi mengenai hal tersebut jauh lebih sulit ditemukan daripada referensi untuk pemrograman itu sendiri.

Tulisan awal ini merupakan versi yang perlu diolah lebih lanjut agar bisa menjadi referensi yang membantu para peserta namun diharapkan bisa memenuhi kebutuhan, sekurangnya sebagai petunjuk, bagaimana soal-soal seleksi olimpiade sains bidang informatika dan komputer ini dibuat. Masukan untuk menyempurnakan tulisan ini sangat diharapkan untuk meningkatkan manfaat terutama para calon peserta dan pembina peserta dalam mempersiapkan diri menghadapi proses seleksi.

Lampiran A Materi Uji

1. Materi Uji Aritmatika

Sekali lagi, walaupun mengambil topik mengenai aritmatika, aspek terpenting dari materi ini bukan pada hitung menghitungnya tetapi pada uji kemampuan analitisnya. Aspek-aspek analitis dalam persoalan aritmatika dijelaskan pada bagian berikut ini.

1.1. Mampu Membentuk Model Aritmatika/Matematika serta melakukan deduksi/induksi Model

Dalam problem solving, seringkali diperlukan tahapan pemodelan masalah yang sebagian menggunakan model matematika/aritmatika dan menyederhana-kannya sehingga menjadi model yang lebih sederhana dan siap dikomputasikan dalam bentuk algoritma. Model yang tidak tepat berakibat pada kegagalan dalam pemecahan masalah.

Contoh:

Uang Amir lebih banyak dari uang Ali. Jika dijumlahkan uang keduanya lebih dari 50 ribu rupiah, sementara selisih uang Amir dengan uang Ali lebih dari 30 ribu rupiah. Berapakah kemungkinan uang Amir yang paling tepat?
Model permasalahan: Uang Amir = x, Uang Ali = y, dan dari deskripsi di atas

- Pers-I: x > y
- Pers-II: x+y > 50000
- Pers-III: |x-y| > 30000

Dari Pers-I dan Pers-III: menghasilkan Pers-IV: x-y > 30000
Dari Pers-II dan Pers-IV: jika dijumlahkan menghasilkan 2x>80000.
Maka, x > 40000

1.2. Memahami Sifat-sifat Bilangan

Untuk sejumlah masalah, sifat-sifat dari bilangan harus dipahami secara logis

Contoh:

Jika n dan p adalah dua bilangan bulat, dan $n + p$ berharga ganjil, manakah dari berikut ini bil ganjil?
(A) $n – p + 1$
(B) np
(C) $n^2 + p^2 – 1$
(D) $3^p + 5^n$
(E) $(p – n)(n – p)$

A bukan, karena $(n+p)$ adalah ganjil maka dari n dan p salah satu ganjil dan yang lain genap. Selisih antara n dan p pasti ganjil sehingga jika ditambah 1

menjadi genap.
B bulan karena perkalian antara suatu bilangan genap dengan bilangan apapun akan menjadi genap.
C bukan karena pangkat bulat positif berapapun dari bilangan genap, tetap genap, dan ganjil tetap ganjil, kemudian ganjil ditambah genap dan dikurang ganjil menjadi genap.
D bukan karena pangkat bulat positif berapapun dari bilangan ganjil tetap bilangan ganjil, dan jumlah dua bilangan ganjil menjadi genap.
E benar, karena perkalian antara dua bilangan ganjil menghasilkan bilangan ganjil.

1.3. Mengkaitkan dengan Konteks Masalah

Konteks dari soal perlu diperhatikan dan konteks tersebut kadang-kadang hanya tersirat saja. Yang dimaksud dengan konteks di sini adalah pemahaman umum akan sesuatu yang sewajarnya diketahui pula.

Contoh:

jika lonceng berdentang setiap 1 detik, dalam jumlah dentang yang sesuai waktu yang ditunjukkan, maka tepat pada pukul berapa dentang terakhir yang menunjukkan jam 6? Apakah pukul 6:00:06?
Salah, seharusnya pukul 6:00:05 karena dentang-dentang tsb pada pukul 6:00:00, pukul 6:00:01, pukul 6:00:02, pukul 6:00:03, pukul 6:00:04 dan pukul 6:00:05!! Konteks disini adalah dentang pertama terjadi pada tepat pukul 6, dan penomoran detik/menit dimulai dari 0, 1, ... dst.

1.4. Memahami Formula Rekursif

Banyak masalah pemodelan dengan tingkat kesulitan yang tinggi atau pemrogramannya sendiri memerlukan pemecahan dengan algoritma rekursif. Pemahaman fungsi rekursif membantu dalam pemahaman memahami bagaimana bekerjanya algoritma rekursif.

Contoh:

Jika didefinisikan $f(n) = n\ f(n–1)$ untuk setiap $n > 0$ dan $f(0) = 1$, maka berapakah $f(10)/(f(7) \times f(6))$?
Pahami perilaku fungsi rekursif tsb, sbb,
$f(n) = n.f(n–1) = n.(n–1).f(n–2) = n.(n–1).(n–2).f(n–3) = ... = n(n–1)(n–2)(n–3)....2.1 = n!$
Sehingga, $f(10) = 10!$ dan $f(7) = 7!$ serta $f(6) = 6!$.
$10!/7! = (10.9.8.7.6.5.4.3.2.1)/(7.6.5.4.3.2.1) = 10.9.8$
Dan $(10.9.8)/(6.5.4.3.2.1) = 1$

1.5. Eksplorasi dalam Masalah Kombinatorik

Dalam problem solving seringkali masalah yang diberikan bersifat kombinatorik (mendapatkan setiap kemungkinan kombinasi/permutasi jawaban). Untuk memecahkannya terkadang seluruh kemungkinan tersebut harus diperiksa untuk

mendapatkan pemecahan yang umum.

Contoh:

Jika diketahui dalam perkalian matriks A (*m*x*n*) dengan B (*n*x*p*) diperlukan biaya *mnp*. Sementara untuk perkalian tiga matriks A.B.C dengan A(*m*x*n*), B(*n*x*p*) dan C(*p*x*q*) ternyata terdapat dua kemungkinan biaya yang bergantung pada urutannya:

- urutan (A.B).C (yaitu A dikali B dahulu kemudian dikali C), dan
- urutan A.(B.C) (yaitu B dikali C dahulu kemudian dikali A).

Urutan (A.B).C memerlukan harga *mnp* + *mpq* sementara urutan A.(B.C) memerlukan harga *npq* + *mnq*. Kedua harga bisa berbeda sesuai dengan harga-harga *m*, *n*, *p*, *q* tsb. Pertanyaannya, untuk perkalian empat matriks A.B.C.D dengan A(10x4), B(4x15), C(15x2), dan D(2x20) manakah urutan dengan biaya minimum?

Kemungkinan-kemungkinan urutan adalah (diperoleh dengan permutasi ketiga tanda perkalian "."):

- urutan (((A.B).C).D), biaya 10x4x15+10x15x2+10x2x20 = 1300
- urutan ((A.B).(C.D)), biaya10x4x15+15x2x20+10x15x20 = 4200
- urutan ((A.(B.C)).D), biaya 4x15x2+10x5x2+10x2x20 = 600
- urutan (A.((B.C).D)), biaya 4x15x2+4x2x20+10x4x20 = 1080
- urutan ((A.B).(C.D)), biaya 15x2x20+10x4x15+10x15x20 = 4200
- urutan (A.(B.(C.D))), biaya 15x2x20 + 4x15x20+10x4x20 = 4200

1.6. Berpikir secara "Cerdas"

Jika menghadapi suatu masalah komputasi yang kelihatannya tidak mungkin, pasti ada sesuatu di balik itu!! Dapatkanlah dengan bantuan pemahaman akan sifat-sifat operasi aritmatika untuk mendapatkan model matematis yang lebih sederhana.

Contoh 1:

Berapa digit terakhir dari 2^{2003}*?* Apakah anda ingin menghitungnya sendiri (secara manual)? Tentu tidak, pasti ada penyederhanaannya. Dengan mengubah n=1, 2, 3, …, dst, perhitungan 2^n menghasilkan deret 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, dst. Amati angka terakhir dari setiap bilangan, kita mendapatkan perulangan dari 6 – 2 – 4 – 8 pada n mod 4 = 0, 1, 2, 3. Jadi jika n=2003, diperoleh 2003 mod 4 = 3, yaitu memiliki digit terakhir 8.

Contoh 2:

Ketiga digit awal dari hasil perkalian 2^{2002} x 5^{2005} jika dijumlahkan adalah? Ini juga tidak mungkin dihitung manual. Perhatikan bilangan dasarnya 2 dan 5 yang jika dikalikan menjadi 10. Karena setiap pasang faktor 2 dan 5 menghasilkan 10 berarti hanya menambah 0 di digit terkanan. Ada 2002 pasang faktor-faktor tsb sehingga 2^{2002} x 5^{2005} = 5^3 x 10^{2002}= 125 10^{2002}. Penjumlahan tiga digit awal 1+2+5=8

Contoh 3:

Hitunglah (80! x 38!) /(77! x 40!).

Menggunakan sifat sbb untuk a dan b bulat positif, a > b, maka a!/b! = a.(a –

1).(a – 2)…(b + 1). Maka:
(80! x 38!) /(77! x 40!) = (80!/77!) / (40!/38!)
= (80x79x78) / (40x39)
= (80/40) x (78/39) x 79
= 2 x 2 x 79 = 316
yang dapat dihitung tanpa kalkulator.

2. Materi Uji Analitika dan Logika

Dalam pemodelan masalah pemrograman selain dengan model aritmatika, juga keterhubungan antara entitas/aspek dalam masalah perlu dipahami secara relational untuk mendapatkan model algoritmis yang lebih akurat. Kemampuan analitis tsb diperlukan dalam menghasilkan model keterhubungan/relasional tsb.

Sayangnya tidak ada metodologi yang sistematik karena pada dasarnya sangat bergantung pada kreatifitas peserta uji. Namun, ada kesamaan umum dalam pemecahan masalahnya, yaitu

- penggunaan model diagram sangat membantu untuk menggambarkan keterhubungannya secara menyeluruh berdasarkan keterhubungan yang fragmental (dari pernyataan-pernyataan terpisah atau asumsi-asumsi yang dibuat),
- keterhubungan itu sendiri seringkali bersifat implisit sehingga perlu pemahaman yang hati-hati dan perlu pemahaman akan gaya bahasa "penceritaannya",
- khususnya untuk asumsi yang dibuat segera dieliminasi jika kontradiksi dengan model diagram, dan
- model diagram yang telah dibentuk perlu diverifikasi (dikaji ulang) dengan pernyatan-pernyataan yang diberikan agar terjaga konsistensi, dan
- Selalu berpikir adanya kemungkinan yang tertinggal atau tersamar yang belum dikaji ke dalam model

Contoh 1:

Terdapat 7 bilangan bulat A, B, C, D, E, F, dan G yang jika diurutkan membentuk deret bilangan cacah berurutan (misalnya 4,5,6,…) dengan pernyataan-pernyataan berikut:
(1) D berharga 3 kurangnya dari A
(2) B adalah angka di tengah jika semua diurutkan
(3) Kurangnya F dari B = kurangnya D dari C
(4) G lebih besar dari F
Jika diurutkan, urutannya adalah?

Untuk memudahkan urutan tsb misalnya [x1–x2–x3–x4–x5–x6–x7]
Dari perny. (2) diketahui x4=B, maka menjadi [x1–x2–x3–B–x5–x6–x7]
Dari perny. (3) F berada di ruas sebelah kiri B (bisa x1, x2 atau x2).
Jika F=x1 maka
D adalah x2 dan C adalah x5 ([F–D–x3–B–C–x6–x7]), atau
D adalah x3 dan C adalah x6 ([F–x2–D–B–x5–C–x7]).
Akan tetapi dari perny. (1) membatalkan kedua kemungk. asumsi ini karena A harus berada 3 posisi di kanan D yang sudah ditempati C.

Jika F=x2 maka
D adalah x1 dan C adalah x3 ([D–F–C–B–x5–x6–x7]), atau
D adalah x3 dan C adalah x5 ([x1–F–D–B–C–x6–x7]), atau
D adalah x5 dan C adalah x7 ([x1–F–x3–B–D–x6–C]).
Akan tetapi dari perny. (1) hanya yang kedua yang mungkin karena yang pertama posisi A = posisi B atau yang ketiga posisi A berada di luar (setelah x7). Untuk sementara [x1–F–D–B–C–A–x7] dicatat sebagai salah satu solusi.

Jika F=x3 maka
D adalah x1 dan C adalah x2 ([D–C–F–B–x5–x6–x7]), atau
D adalah x5 dan C adalah x6 ([x1–x2–F–B–D–C–x7]), atau
D adalah x6 dan C adalah x7 ([x1–x2–F–B–x5–D–C]).
tetapi dari (1) semua tidak mungkin (yang pertama posisi A = posisi B, kedua yang lain posisi A ada di luar).
Jadi ternyata hanya tinggal satu kemungkinan yaitu [x1–F–D–B–C–A–x7].
Dari perny. (4) diperoleh G=x7, sehingga diperoleh juga E=x1. Hasilnya diketahui urutannya adalah E, F, D, B, C, A, G

Contoh 2:

Delegasi-delegasi dari negara W dan negara R duduk berhadap-hadapan pada meja perundingan. Masing-masing delegasi terdiri atas seorang ketua, dua atase militer dan dua wakil kamar dagang negara masing-masing. Delegasi W beranggotakan A, B, C, D, dan E. Delegasi R beranggotakan F, G, H, I, dan J. Masing-masing delegasi berada pada sisi-sisi memanjang berlainan (satu negara pada sisi yang sama dan ketua duduk di tengah delegasinya). Batasan dalam mengatur urutan duduk mereka:
(1) Delegasi W menempatkan A dan B di kedua ujung barisannya.
(2) Kuping kanan G tuli shg ia harus paling kanan dari delegasi R.
(3) Baik D maupun F bukan ketua.
(4) Para atase militer W, salah seorangnya B, didudukkan berdampingan, dan tidak ada satupun yang berseberangan dengan atase militer R
(5) G bukan atase militer.
(6) C wakil dari kamar dagang, duduk berseberangan dengan H.

Manakah yang paling mungkin mengenai F berikut?
(A) Wakil kamar dagang yang duduk di sebelah I
(B) Wakil kamar dagang yang duduk di sebelah H
(C) Wakil kamar dagang yang duduk berseberangan dengan B
(D) Atase militer yang duduk di sebelah I
(E) Atase militer yang duduk di sebelah J

Seperti pada contoh sebelumnya, dibuat diagram sbb
x1–x2–x3–x4–x5 negara W
y1–y2–y3–y4–y5 negara R
Dari (1) kemungkinan {x1,x5} adalah {A,B} atau {B,A}
Dari (2) maka y5=G yang karena pernyataan (4) dan (5) (G bukan a.m dan B adalah a.m) menyebabkan x5=B, sehingga (atase militer dengan bold)

A –x2–x3–**x4 –B**
y1–y2–y3–y4–G

Dari pernyataan (6) dan (4) diperoleh C = x2 dan y2 = H, sehingga
A –C–x3–**x4 –B**
y1–H–y3–y4–G

Dari pernyataan (3) dan diagram di atas D = x4 dan F = y1 atau y4
A –C –E **–D –B**
y1–H –y3–y4–G

Jadi tinggal 2 kemungkinan F=y1 (atase militer), atau F=y4 (wakil kamar dagang). Jika atase militer maka (D) dan (E) salah karena sebelah y1 adalah H. Jika wakil kamar dagang maka (B) salah karena H atase militerdan (C) salah karena B ada di depan G. Jadi tinggal pilihan (A) yang paling mungkin. (Note: ini bukan satu-satunya kemungkinan. Kemungkinan lainnya masih ada tapi tidak ada di kelima pilihan itu).

3. Materi Uji Algoritmika

Sebagaimana dalam penjelasan awal, materi uji algoritmika adalah selain menguji kemampuan dalam memahami suatu algoritma yang diberikan, juga menguji kemampuan merancang suatu algoritma pemecahan masalah yang diberikan. Namun, untuk tingkat OSK/OSP pada saat ini belum memungkinkan untuk dilakukan terutama terkendala masalah teknis pemeriksaan kebenaran jawabannya. Kemampuan dalam perancangan tersebut baru akan diujikan kemudian di tingkat OSN.

Karena pada tingkat OSK/OSN ini peserta harus dapat memahami algoritma yang diberikan maka hal yang penting untuk dikuasai adalah penulisan notasi algoritma yang digunakan oleh soal-soal. Penulisan algoritma mungkin menggunakan suatu cerita (bahasa sehari-hari) atau mengikuti notasi/tatacara yang didefinisikan sebagai pseudopascal5. Proses dari algoritma umumnya bersifat prosedural/imperatif saja6.

Aspek-aspek yang biasanya diujikan adalah:

1. penggunaan variabel (berarti sifat-sifatnya juga) dalam kaitannya dengan proses algoritma tetapi tidak berkaitan dengan sifat variabel yang spesifik pada bahasa pemrograman tertentu[7].

[5] Lebih tepatnya, "TOKI Pseudopascal" dengan dokumen yang diposting di webite dengan URL di http://www.toki.or.id/toki2006/pseudopascal.pdf

[6] Hingga IOI 2006 soal-soal yang diujikan masih mementingkan efisiensi dalam problem solvingnya sehingga rancangan yang object-oriented ataupun deklaratif belum perlu (atau malah tidak disarankan demi efisiensi solusi) untuk digunakan. Bahasa pemrograman yang populer di IOI adalah FreePascal, C dan C++. Khususnya C++ digunakan terutama untuk memudahkan sejumlah codingnya saja, bukan karena aspek object-orientednya.

[7] Dalam bahasa C terdapat kerumitan definisi mengenai array yang tidak terjadi dalam bahasa Pascal. Dalam bahasa Java character yang digunakan dalam String adalah unicode (16 bit) sementara dalam bahasa C atau Pascal adalah byte (8 bit). Dalam bahasa Pascal terdapat variabel tipe string standard pascal yang byte ke-0 berisi panjang logical string di dalamnya sementara dalam C variabel String adalah array character dengan indeks dari 0. Dalam bahasa Pascal array bisa berindeks suatu range bilangan bulat apa saja termasuk bulat negatif, juga dapat menggunakan indeks non numerik.

2. aliran kendali proses[8]: blok **begin-end**, pecabangan **if-then**, pencabangan **if-then-else**, pencabangan **case-option**, loop **while-do**, loop **repeat-until**, dan loop **for**.
3. ekspresi lojik dengan operator lojik **and**, **or**, **xor**, **not**,
4. pemanggilan prosedur dan fungsi dan
5. aliran proses dari algoritma (prosedur atau fungsi) rekursif
6. struktur array (satu dimensi atau lebih)

Contoh-contoh

1. Diberikan fungsi berikut

```
function apaini(a: integer; b: integer): integer;
var x,y,r: integer;
begin
   x := a;
   y := b;
   while (y <> 0) do
   begin
      r := x mod y;
     x := y;
     y := r;
   end;
   apaini := x;
end;
```

Pertanyaan: Jika fungsi tsb dipanggil dengan "writeln(apaini(414, 662));" berapakah yang dicetaknya?

Pembahasan: Pemanggilan tsb akan dijalankan dengan variabel a mula-mula berharga 414 dan b berharga 662. Kedua variabel dalam algoritma tidak mengalami perubahan apapun, jadi fungsinya hanya menyampaikan harganya ke variabel x dan y masing-masing. Dalam fungsi tersebut terdapat struktur loop while-do dengan variabel yang aktif (berubah-ubah) dalam loop tersebut bernama x dan y. Terdapat variabel lain yaitu r yang berfungsi sebagai variabel pembandu operasi. Dalam struktur begin-end di dalam loop while-do tsb terjadi perubahan harga x diganti dengan y dan y diganti dengan harga r yang sebelum x dan y berubah r diisi x mod y. Jadi algoritma ini saling memodulokan dua bilangan. Dalam memahami loop while-do, penting diperhatikan inisialisasi dan kondisi iterasi berakhir. Inisialisasinya adalah mengisi variabel x dengan 414 dan y dengan 662. Iterasi akan berakhir apabila y sebagai variabel yang akan memodulokan x berharga 0. Jadi urutannya:

414 mod 662 = 414
662 mod 414 = 248
414 mod 248 = 166
248 mod 166 = 82
166 mod 82 = 2
82 mod 2 = 0

Iterasi berhenti karena y berikutnya berharga 0. Terminasi algoritma mengembalikan harga x yaitu 2 kepemanggil fungsi. Terakhir, pemanggil fungsi melakukan pencetakan harga yang diperoleh dari pemanggilan tersebut yaitu 2.
Jadi jawabnya adalah 2.

[8] Dalam edisi pseudopascal yad sedang dipertimbangkan struktur *exception handling* **try-catch**. Untuk edisi sekarang belum termasuk.

2. Diberikan fungsi berikut

```
function apaitu(a: integer; b: integer): integer;
begin
  count := count + 1;
   if (a > b) then apaitu := apaitu(b, a)
   else  if (a = 0) then apaitu := b
      else  apaitu := apaitu (b mod a, a)
end;
```

Pertanyaan: Jika fungsi tsb dipanggil dengan "writeln(apaitu(1001, 1331));" berapakah yang dicetaknya?

Pembahasan: Fungsi tersebut adalah fungsi rekursif dengan nama **apaitu**. Itu ditandai adanya pemanggilan fungsi bernama sama dengan dirinya. Di dalam fungsi ada struktur bersarang (nested structure) if-then-else membentuk 3 pencabangan.

- Cabang pertama, jika harga dalam variabel a lebih besar dari b, algoritma akan melakukan pemanggilan rekursif dengan penukaran posisi variabel a menjadi b dan b menjadi a.
- Cabang kedua, jika a berharga 0 maka akan dikembalikan harga b.
- Cabang ketiga, tentu untuk a lebih kecil dari b, akan memanggil secara rekursif dengan posisi harga a diberi harga (b mod a) dan posisi harga b diberi harga a.

Untuk semua cabang, tidak ada operasi lain sehingga hasilnya langsung dikembalikan ke pemanggil sebelumnya. Operasi pada variabel count tidak berarti apa-apa berkenaan dengan pertanyaan ini.
Pemanggilan apaitu(1001, 1331) akan membawa ke cabang ketiga lalu memanggi apaitu(330, 1001). Kemudian akan membawa ke cabang ketiga lagi lalu memanggil apaitu(11, 330). Kembali lagi ke cabang ketiga dan memanggil apaitu(0, 11). Pemanggilan terakhir ini akan memberikan harga variabel b menjadi harga yang dikembalikan dari fungsi tersebut, kemudian diteruskan ke pemanggilan sebelumnya, hingga ke pemanggilan pertama. Jadi jawabannya 11.

Catatan: Jika anda dapat memahami algoritma-algoritma pada kedua contoh di atas dengan baik, maka contoh ini dan contoh di atas menghasilkan harga fungsi yang tepat sama.

3. Diberikan fungsi berikut

```
if (a and b) or ((not c) and d) then
   if ((a or not b) and c) or (b and (not a)) then
      writeln(1)
   else
      if (a or (d and b)) and (not b) then
         writeln(2)
      else
         writeln(4)
else
   if not (d and c) and (not a) then
      writeln(5)
   else
      writeln(6);
```

Pertanyaan: Jika dijalankan dan ternyata mencetakkan harga 4 maka urutan harga-harga a, b, c, d yang mungkin adalah?
(A) TRUE, FALSE, TRUE, FALSE
(B) TRUE, TRUE, TRUE, FALSE
(C) FALSE, FALSE, TRUE, TRUE
(D) TRUE, TRUE, FALSE, FALSE
(E) TRUE, FALSE, FALSE, TRUE

Pembahasan: Untuk soal ini pilihan harus diujikan pada setiap ekspresi boolean/lojik dalam algoritma di atas. Untuk memudahkan kita sebut E1, E2, E3 dan E4 untuk keempat ekspresi lojik di atas dimulai dari ekspresi paling atas. Pilihan (A): E1 berharga FALSE, lalu memeriksa E4 yang juga FALSE sehingga menjalankan writeln(6).
Pilihan (B): E1 berharga TRUE, lalu memeriksa E2 dan juga TRUE, kemudian menjalankan writeln(1).
Pilihan (C): E1 berharga FALSE, lalu memeriksa E4 yang berharga TRUE, kemudian menjalankan writeln(5).
Pilihan (D): E1 berharga TRUE, lalu memeriksa E2 yang berharga FALSE, kemudian E3 berharga FALSE kemudian menjalankan writeln(4). Jadi untuk mencetak 4, maka pilihan D yang benar.
Untuk cross-check dan masih ada waktu dalam melakukannya, pilihan (E) bisa diperiksa: E1 berharga TRUE lalu memeriksa E2 yang berharga FALSE dan kemudian E3 TRUE sehingga menjalankan writeln(2).

Catatan: untuk pemeriksaan ekspresi lojik, jika cukup berlatih maka pemeriksaan tersebut bisa dipercepat dengan hanya memeriksa sebagian dari ekspresi. Misalnya operator **or** hanya mensyaratkan salah satu dari kedua operand yang TRUE menyebabkan ekspresi tersebut TRUE, sebaliknya jika salah satu operand dari operator **and** berharga FALSE maka ekspresi tersebut menjadi FALSE.

4. Suatu robot berdasarkan harga *a* bilangan positif yang diberikan, akan menjalankan sederetan perintah berikut (algoritma dengan penulisan bahasa sehari-hari):

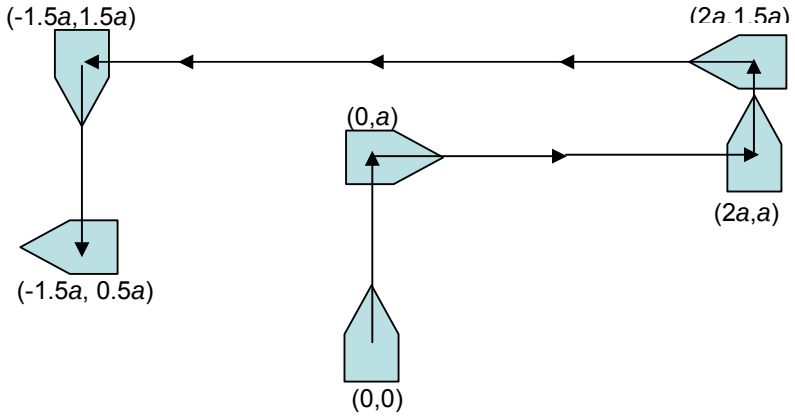
```
begin
        melangkah dengan jarak a ke depan,
        memutar arah ke kanan tegak lurus,
        melangkah sepanjang 2a,
        memutar ke arah kiri tegak lurus,
        melangkah sepanjang ½ a,
        memutar ke arah kiri tegak lurus,
        melangkah sepanjang 3½ a,
        memutar ke arah kiri tegak lurus,
        melangkah sepanjang a.
        memutar ke arah kanan tegak lurus.
end;
```

Pertanyaan: jika posisi awal ada di (0, 0) dan robot sedang menghadap ke arah sumbu-y positif, dengan *a* = 2 maka posisi akhir robot adalah ?

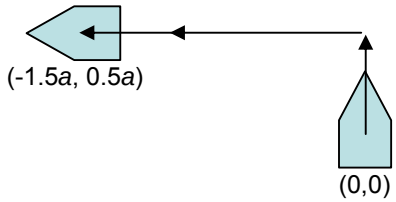
Pembahasan: Perhatikan diagram lintasan tsb. Ternyata, robot pada saat

awal di (0,0) menghadap ke sumbu-y, setelah menjalani lintasannya akan berada di (-1.5*a*,0.5*a*) dan menghadap ke kiri (270^o) dari semula. Dengan *a*=2 maka akan berada di (-3,1).

Deretan langkah di atas digambarkan pada Gambar 1 ternyata efeknya sama saja dengan hanya melakukan langkah-langkah seperti pada Gambar 2. Jadi selanjutnya, cukup memperhatikan kondisi awal dan kondisi akhir tersebut, kemudian putarkan ke kanan 90^o.



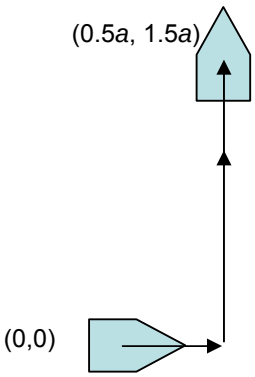
Gambar 1



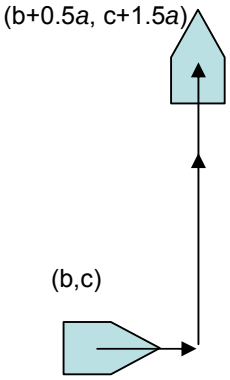
Gambar 2

Pertanyaan: Sekarang dengan pertanyaan baru: jika posisi awal ada di (0, 0) dan robot sedang menghadap ke arah sumbu-x positif, dengan *a* = 4 maka dimanakah posisi akhirnya?

Pembahasan: Dengan memutarkan Gambar 2 maka diperoleh posisi akhir di (0.5*a*, 1.5*a*) seperti terlihat pada Gambar 3. Dengan *a* = 4, posisi menjadi menjadi (2, 3) dan menghadap ke sumbu-y positif.



Gambar 3



Gambar 4

Pertanyaan: Kalau posisi awal bukan di (0,0) melainkan di (*b*, *c*) maka dimanakah posisi akhir tsb?

Pembahasan: Sederhana saja seperti terlihat pada Gambar 4, tinggal geser posisi awal dari (0,0) ke (*b*, *c*), menjadi (*b*+0.5*a*, c+1.5*a*).

Pertanyaan: Jika posisi awal ada di (0, 0) dan robot sedang menghadap ke arah sumbu-x positif, deretan perintah tersebut dilakukan secara berulang sebanyak 2 kali (3 kali, 4 kali, dst) maka posisi akhir robot adalah?

Pembahasan: Jadi ingat untuk selalu memperhatikan posisi awal dan akhir saja seperti pada Gambar 2:

- Pertama: arah sumbu-x positif, posisi (0,0) berhenti di (0.5*a*, 1.5*a*), dengan arah sumbu-y positif
- Kedua: arah sumbu-y positif, posisi (0.5, 1.5*a*) berhenti di ……, dengan arah ……
- Ketiga: arah ……..., posisi …… berhenti di ……, dengan arah ……

Pertanyaan: Jika posisi awal ada di (0, 0) dan robot sedang menghadap ke arah sumbu-*x* positif, deretan perintah tersebut dilakukan berulang sebanyak 3 kali dengan harga *a* pertama = 2 , harga *a* kedua = 4 dan harga *a* ketiga = 1. Dimanakan posisi akhir robot?

Pertanyaan: Jika posisi akhir ada di (0,0) dengan robot sedang menghadap ke arah sumbu-*y* positif setelah deretan perintah tersebut dilakukan berulang sebanyak 5 kali dengan *a*=4, berada di manakah robot itu sebelumnya?

Pembahasan: Silakan mencoba menjawab kedua pertanyaan tersebut.

4. Materi Uji Programming

Pada tingkat OSN peserta akan diuji dalam materi pemrograman yang sebenarnya. Selain peserta harus menguasai pemrograman dengan bahasa yang digunakan

yaitu FreePascal atau C atau C++, tetapi juga peserta harus mampu melakukan problem solving itu sendiri. Soal-soal yang diberikan bukan berupa soal spesifikasi pemrograman yang eksplisit, namun permasalahan yang mana pemrograman hanya sebagai alat dalam pemecahan masalah itu sendiri. Kemampuan analitikal peserta diperlukan karena program yang nantinya dituliskan akan dijalankan dengan masukan yang sangat ekstrim sehingga dengan cara yang naif (tanpa analisis mendalam) program peserta tidak akan dapat berjalan dalam batas waktu yang diberikan. Aspek pertandingan lainnya adalah bahwa peserta harus mengerjakan soal ini dalam waktu yang terbatas pula!

Permasalahan: buatlah program yang dapat menghitung berapa banyak 0 di belakang bilangan *N*! dengan harga *N* yang sangat besar sekali (misalnya 10^{30}) ?

Pembahasan: Jika anda menjawabnya dengan memperkalikan semua bilangan *N*.(*N*-1).(*N*-2)....2.1 dst maka anda hanya berhasil menjawab untuk *N* yang kecil (sebatas ukuran harga terbesar dari bilangan bulat yang disediakan serta batas waktu komputasi yang diberikan).

Jadi anda perlu melakukan analisis sbb. Karena banyaknya angka nol di belakang suatu angka bergantung pada berapa banyak kemunculan faktor 2 dan faktor 5 (mengapa? karena 2 kali 5 adalah 10). Dan khususnya, untuk suatu bilangan *N*! dapat dibuktikan banyaknya kemunculan faktor 5 tidak akan lebih banyak dari banyaknya kemunculan faktor 2. Maka, pertanyaan di atas dapat dijawab dengan hanya menghitung total banyaknya kemunculan faktor 5 dari bilangan-bilangan pembentuk *N*! Bilangan yang berisi faktor 5 adalah bilangan-bilangan kelipatan 5 saja jadi cukup kita memperhatikan bilangan-bilangan kelipatan 5 ini.

Misalnya dalam 10! hanya ada dua bilangan yang memiliki faktor 5 yaitu 5 dan 10 sendiri sehingga. Contoh lain, dalam 29! ada 5, 10, 15, 20, 25 yang berisi faktor 5, dan karena pada 25 faktor 5 muncul dua kali menyebabkan 29! berisi kemunculan faktor 5 sebanyak 6 kali (jadi ada 6 nol di belakang 29!). Dengan mengiterasi *i* dari 5 ke *N* dengan kelipatan 5, dan mendapatkan berapa banyak faktor 5 dari bilangan i, lalu menjumlahkannya, maka anda dapat menjawabnya dengan baik untuk level OSN!

```
i := 5;
count := 0;
while i <= N do
begin
   // menghitung berapa banyak kemunculan faktor 5 dalam i
   j := i;
   while (j mod 5) = 0 do begin
      j := j div 5;
      count := count + 1;
   end;
   i := i + 5;
end;
```

Untuk tingkat OSN ini dengan cara demikian anda dapat memperoleh nilai penuh. NAMUN, untuk tingkat lebih lanjut mungkin harga *N* bisa diberikan jauh lebih besar misalnya $N = 10^{12}$, algoritma harus beriterasi luar (while) sebanyak 2.10^{11} kali -- dan untuk setiap harga *i* akan diperlukan sekian kali lagi beriterasi untuk menghitung banyaknya faktor 5 dalam *i* yang akhirnya akan memerlukan waktu komputasi yang besar!. Apalagi jika waktu eksekusi lebih dibatasi jelas solusi tsb tidak akan

memberikan nilai penuh. Untuk itu anda harus menggali ide lebih lanjut lagi sbb. Perhatikan bahwa:

- bilangan-bilangan berfaktor 5 adalah semua bilangan kelipatan 5. Jika J_1 adalah jumlah bilangan kelipatan 5 yang ≤ N tersebut maka $J_1 = N$ **div** 5.
- di antara bilangan-bilangan itu terdapat bilangan-bilangan kelipatan 25, yaitu yang menyumbang faktor 5 sebanyak dua kali. Jika J_2 adalah banyaknya kemunculan bilangan kelipatan 25 yang ≤ N, maka $J_2 = N$ **div** 25.
- di antara bilangan-bilangan itu terdapat bilangan-bilangan kelipatan 125, yaitu yang menyumbang faktor 5 tiga kali. Jika J_3 adalah banyaknya kemunculan bilangan kelipatan 125 yang ≤ N maka $J_3 = N$ **div** 125.
- ... dst.

Maka, jumlah faktor 5 pada $N!$ = $J_1 + J_2 + J_3 + ...$ = (N **div** 5) + (N **div** 25) + (N **div** 125) + ... berdasarkan analisis ini anda cukup membuat iterasi untuk menghitung dan mentotalkan (N **div** i) dengan i deret 5, 25, 125, ... selama $i \le N$. Algoritma yang diperoleh hanya berisi 8 baris saja sebagai berikut. Untuk $N = 10^{12}$, iterasi hanya dilakukan kurang dari 18 kali ($\log_5(10^{12}) < 18$). Jadi program yang efisien untuk menjawab masalah di atas adalah program yang sangat pendek sebagai berikut.

```
readln(N);
i := 5;
count := 0;
while i <= N do begin
   count := count + (N div i);
   i := i * 5;
end;
writeln(count);
```

Dengan contoh ini, hendak ditunjukkan bahwa masalah yang diberikan dalam materi pemrograman bukanlah semata masalah pemrograman itu saja melainkan masalah analitika yang mendalam kemudian pemrograman hanyalah sebagai realisasinya yang seringkali hanya dengan pembuatan program yang singkat/pendek semata.

Pada tingkat kesulitan yang lebih tinggi hingga tingkat Olimpiade Internasional, berlaku cara berpikir demikian. (Lihat pembahasan sejumlah soal contoh di website http://www.toki.or.id).

Lampiran B Contoh Soal OSK/OSP

1. Deret bilangan antiFibonacci didefinisikan secara rekursif sbb.:

$$\begin{cases} f_1 = 1 \\ f_2 = 0 \\ f_n = f_{n-2} - f_{n-1} \quad n > 2 \end{cases}$$

Dengan mengambil satu harga *n* kemudian Anda menjumlahkan bilanganbilangan tersebut mulai darif1 s.d.fnmaka berapakahnterkecil agar jumlah itu > 300?

a. 9
b. 10
c. 14
d. 17
e. 20

2. Jika semua W adalah X, semua X adalah Y, dan semua Y adalah Z, maka manakah yang tidak benar?

a. Semua X adalah Z
b. Semua W adalah Y
c. Semua W adalah W
d. Semua W adalah Z
e. Semua Y adalah W

3. Suatu tim peneliti yang terdiri atas lima orang dipilih dari antara empat matematikawan A, B, C, dan D serta empat fisikawan, E, F, G dan H. Paling sedikit tiga matematikawan harus dipilih. Tetapi kesulitannya:

A tidak mau bekerja dengan D
B tidak mau bekerja dengan E
F tidak mau bekerja dengan G
D tidak mau bekerja dengan F

Jika B dipilih, siapa yang pasti juga akan dipilih?

a. F
b. G
c. A
d. C
e. D

4.
```
  SEND
 MORE +
MONEY
```

S, E, N, D, M, O, R, dan Y masingmasing mewakili satu digit integer (bilangan bulat) positif; dan masingmasing mewakili bilangan yang berbeda. S dan M tidak sama dengan 0.

Berapakah nilai N+O+S+E?

a. 20
b. 16
c. 23

d. 7
e. 11

5. Perhatikan potongan program berikut

```
n:=10;
x:=0;
for i:=0 to n do
begin
      x:=x+2*i;
end;
writeln(x);
```

Berapakah output dari program di atas?

a. 100
b. 112
c. 90
d. 110
e. 72

6. Perhatikan tahapan-tahapan berikut:
Misalkan ada dua variable *x* dan *y*, dan variable hasil yang nilai awalnya 0. Lakukan proses berikut hingga nilai *x* saat ini lebih besar dari 0:
    1. Jika nilaixgenap maka nilai *hasil := hasil + y*.
    2. Nilaixselanjutnya adalah nilai *x* sebelumnya dibagi dua, bila ada hasil pecahan, maka pecahannya dibuang. (contoh bila nilai *x* sebelumnya 3, maka nilai *x* selanjutnya 1)
    3. Nilai *y* selanjutnya adalah nilai *y* sebelumnya dikali tiga

Bila nilai awal *x* = 10 dan nilai awal *y* = 5, maka nilai akhir variable hasil adalah:

a. 0
b. 5
c. 25
d. 50
e. Salah semua

Lampiran C Contoh Soal OSN

Kata Spiral

Soal OSN 2005

KodeSoal:spiral
Batas Runtime:1 detik / testcase
BatasMemori:1 MB
Masukan:Standard input
Keluaran:Standard output

Deskripsi
Suatu sistem sandi menyandikan kalimat yang diberikan dalam bantuk spiral. Penyusunan tersebut dilakukan membentuk matriks spiral yang dimulai pusat matriks 1 karakter pertama, lalu 1 karakter berikutnya ke kanan, lalu 1 karakter berikutnya ke bawah, lalu 2 karakter berikutnya ke kiri, lalu 2 karakter berikutnya ke atas, 3 karakter berikutnya ke kanan, 3 karakter berikutnya ke bawah, dan seterusnya hingga semua karakter dalam kalimat termasuk dalam spiral. Khususnya, karakter spasi di ganti dengan "_" (underscore), dan jika ada baris/kolom tersisa setelah karakter terakhir maka elemenelemen matriks diisi juga dengan "_" (underscore) tsb. Misalnya kalimat "Seluruh peserta OSN bidang komputer harus mengerjakan soalsoal sebaikbaiknya untuk mendapatkan peringkat terbaik." Dikodekan kedalam matriks sebagai berikut.

```
b a i k . _ _ _ _ _ _
r a i k n y a _ u n t
e b m e n g e r j a u
t - _ b i d a n g k k
_ k s _ h _ p e _ a _
t i u N u S e s k n m
a a r S r u l e o _ e
k b a O _ a t r m s n
g e h _ r e t u p o d
n s _ l a o s - l a a
i r e p _ n a k t a p
```

Masukan
Program membaca satu baris teks paling panjang 250 karakter.

Keluaran
Program harus menghasilkan sejumlah baris sesuai dengan matriks yang dibentuk. Setiap baris keluaran berisikan karakterkarakter dari baris yang sama berturutturut dari kolom paling kiri ke paling kanan tanpa pemisahan (karakterkarakter dituliskan bersambungan menjadi satu string serta jangan lupa setiap spasi menjadi underscore).

Contoh1
Masukan (tertulis dalam satu baris)

```
Seluruh peserta OSN bidang komputer harus mengerjakan soal-soal sebaik-baiknya untuk mendapatkan peringkat terbaik.
```

Keluaran

```
baik._______
raiknya_unt
ebmengerjau
t-_bidangkk
_ks_h_pe_a_
tiuNuSesknm
aarSruleo_e
kbaO_atrmsn
geh_retupod
ns_laos-laa
irep_naktap
```

Contoh2
Masukan (tertulis dalam satu baris)

```
TOKI
```

Keluaran

```
TO
IK
```

Contoh3
Masukan (tertulis dalam satu baris)

```
OSN
```

Keluaran

```
OS
_N
```

Contoh4
Masukan (tertulis dalam satu baris)

```
Bisakah Kamu?????
```

Keluaran

```
_h_Ka
_aBim
_kasu
?????
```

Lampiran D Contoh Soal Pelatnas

Perjamuan Alakapuri

Soal Pelatnas 2006

Nama Soal : Perjamuan Alakapuri
Batasan Memori : 32 MB
Batasan Waktu : 1 s
Input : standard input
Output : standard output

Alkisah, bertujuan untuk memamerkan kekayaannya sekaligus meningkatkan pengaruhnya di kalangan para dewa, dewa kekayaan Kubera berencana mengundang para dewa untuk mengikuti jamuan makan di kotanya, Alakapuri.

Kubera sangat tahu tabiat para dewa yang lain. Para dewa sangat tidak suka diduakan, oleh sebab itu, Kubera harus mengatur jadwal perjamuannya sedemikian rupa sehingga pada suatu saat tertentu, dia hanya menjamu satu dewa saja.

Dengan bantuan para pembantunya, Kubera telah menyusun sebuah daftar yang berisi: nama dewa, waktu di mana dewa tersebut bersedia menghadiri perjamuan, dan perkiraan lama waktu makan dewa tersebut.
Karena Kubera tidak pilih-pilih dalam mengundang para dewa, untuk membuat daftar lebih ringkas, Kubera kemudian berinisiatif mengganti kolom nama dewa menjadi nomor urut saja.

Hasilnya, Kubera mempunyai sebuah tabel yang kira-kira berisi seperti ini:

Dewa Ke (i)	Waktu Awal Jamuan (S)	Durasi (D)
1	2	5
2	9	7
3	15	6
4	9	3

Jelas bahwa Kubera mungkin tidak dapat mengundang semua dewa yang ada karena bisa saja jadwal di mana satu dewa bersedia dijamu beririsan dengan jadwal dewa yang lain. (Kubera tidak mungkin mengusir seorang dewa di tengah-tengah perjamuan. Satu perjamuan berakhir ketika dewa yang sedang dijamu selesai makan, dan lama waktu makan dewa tersebut tidak akan lebih lama daripada waktu perkiraan yang telah dibuat pembantu Kubera)

Contohnya, mengacu pada daftar di atas, dia tidak mungkin mengundang keempat dewa yang ada (karena jadwal dewa 2 dan 3 beririsan sehingga hanya salah satu dari mereka yang bisa diundang). Anggap saja, yang diundang adalah dewa 1, 3, dan 4. Dari satuan waktu ke-2 sampai 6 (hingga sesaat sebelum waktu ke-7), Kubera akan menjamu dewa 1. Kemudian pada waktu ke-9, dia akan mulai menjamu dewa 4 sampai waktu ke-11. Dan terakhir, ia menjamu dewa 3 pada waktu 15 – 20.

Sesuai dengan tujuannya semula, Kubera tentu ingin menjamu sebanyak-banyaknya dewa. Sebagai ahli hitung terbaiknya, Anda diminta oleh Kubera untuk membantunya menghitung berapa banyak dewa yang dapat ia undang ke perjamuannya.

Input

Baris pertama masukan berisi sebuah integer N, menunjukkan banyaknya dewa yang ada. Baris 2, 3, …,N+1 mendeskripsikan jadwal ke-N dewa tersebut. Baris ke i+1 berisikan dua buah integer Si dan Di, yang merepresentasikan waktu di mana dewa ke-i bisa hadir serta perkiraan durasi perjamuan untuk dewa tersebut.

Output

Terdiri atas sebuah integer M, yaitu banyak maksimum dewa yang dapat dijamu.

Contoh Input	Contoh Output
4 2 5 9 7 15 6 9 3	3

Penjelasan Input/Output

Seperti yang diuraikan pada deskripsi soal.

Batasan

N<= 100 000, 1 <=Si<= 1 000 000, 1 <=Di<= 1 000

Catatan: soal ini merupakan soal simulasi pada pelatnas 16 besar TOKI 2006

Lampiran E Daftar Bacaan & Refensi

Bacaan untuk Pra OSN

- Suryana Setiawan. 2006. *Pseudopascal (Versi Olimpiade Sains Bidang Informatika/Komputer)*. TOKI Pusat. (file pseudopascal.pdf) atau bisa didownload di :
  - http://www2.toki.or.id/toki2006/pseudopascal.pdf

- Inggriani Liem - Tim Pembina TOKI. *Aspek Pedagogi Pengajaran Pemrograman Pertama menggunakan Bahasa Pascal*. TOKI Pusat.

- Yohanes Nugroho - Tim Pembina TOKI. *Konsep Dasar Pemrograman Prosedural*. TOKI Pusat.

- Buku-buku Pemrograman Pascal, terutama bagian mengenai:
  - Deklarasi
    - Tipe data
    - Variabel dan konstanta
  - Operasi assignment
  - Operasi aritmetika
  - Ekspresi logika
  - Struktur kontrol
    - Sequence
      - begin—end
    - Analisis kasus (branching)
      - if—then
      - if—then—else
      - case—of
    - Perulangan (loop)
      - while—do
      - repeat—until
  - Fungsi/prosedur
    - Urutan eksekusi
    - Parameter/argumen
    - Rekursivitas

  Seperti contoh:
  - Windra Swastika dan Fauzan Joko. *Referensi Pemrograman Bahasa Pascal*.

- Tim Pembina TOKI. *Contoh Soal dan Pembahasan*. TOKI Pusat.

- Tim Pembina TOKI. *Contoh Soal Seleksi Tertulis (Seleksi Awal)*. TOKI Pusat, bisa didownload di :.
  - http://www2.toki.or.id/download/Analitik.zip
  - http://www2.toki.or.id/download/pascal.zip

- Tim Pembina TOKI. *Contoh Materi Pelatihan Jarak Jauh Pra OSN*. TOKI Pusat.

Bacaan untuk Pasca OSN menuju Olimpiade Internasional

- USACO Training Program
  - http://train.usaco.org

- Steven Halim. *Data Structures & Algorithm*. CS—NUS.
  - http://www.comp.nus.edu.sg/~stevenha/programming/progr amming.html

- Steven S. Skiena, Miguel Revilla. 2003. *rogramming Challenges*. SpringerVerlag.

- Niklaus Wirth. 1976. *Algorithm + Data Structures = Programs*. PrenticeHall.

- Anany Levitin. 2003. *Introduction to the Design and Analysis of Algorithms*. AddisonWesley.

- Richard Neapolitan, Kumarss Naimipour. 1996. *Foundations of Algorithms.* D.C. Heath and Company.

- Robert Sedgewick. 1988. *Algorithms.* AddisonWesley.

- Steven S. Skiena. 1998. *The Algorithm Design Manual.* Springer Verlag.

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. 1989. *Introduction to Algorithms*. The MIT Press.