**Soal 2**

Jelaskan 3 (tiga) kategori persoalan *computing*, yaitu: *tractable*, *intractable*, dan *solvable approximately, but not optimally*. Sertakan juga contoh persoalan *computing*pada masing-masing kategori persoalan tersebut.

**Soal 3**

Sebutkan dan jelaskan 3 (tiga) bidang baru yang muncul pada dunia *computing*di era revolusi industri keempat. Berikan juga contoh implementasi atau pemanfaatan untuk masing-masing bidang tersebut dalam kehidupan sehari-hari.

**Soal 4**

Sebutkan dan jelaskan 2 (dua) cara untuk mengukur kecepatan dari sebuah algoritma.

**Soal 5**

Akan diadakan suatu kompetisi memancing yang di dalamnya, setiap peserta berusaha memancing ikan dengan umpan yang disediakan panitia. Penilaian kompetisi dilakukan dengan membandingkan poin tiap peserta yang didapatkan dari konversi berat ikan yang berhasil ditangkap ke empat kategori nilai. Kategori tersebut ialah:

* Gold (>=1500 gram): 100 poin
* Silver (750-1499 gram): 50 poin
* Bronze (10-749 gram): 25 poin
* Zonk (1-9 gram): 0 poin

Setiap peserta dihitung nilai akhirnya berdasarkan banyaknya ikan yang berhasil ditangkap menggunakan sebanyak 20 umpan yang disediakan panitia. Peserta dilarang menangkap ikan menggunakan umpan lain.

Anda diminta membuat sebuah program untuk melakukan penghitungan nilai akhir satu peserta yang nantinya bisa digunakan untuk dibandingkan dengan peserta lain. Program harus dibuat dengan ketentuan sebagai berikut:

1. Program menerima masukan jumlah tangkapan total (integer). Jumlah ini harus dipastikan valid terlebih dahulu, yaitu bernilai 0-20, sebelum program dapat lanjut ke tahapan berikutnya.
2. Program menerima masukan berat ikan (integer) sebanyak jumlah tangkapan total pada poin 1, kemudian melakukan penghitungan nilai melalui:
   * Konversi berat ke poin. Asumsikan nilai berat sudah pasti valid (>0 gram).
   * Akumulasi/total nilai poin.
3. Program menampilkan akumulasi nilai hasil penghitungan satu peserta dari poin 2.

Buatlah program berdasarkan ketentuan tersebut dengan dekomposisi yang modular dan menggunakan skema standar dengan baik. Jika Anda membuat fungsi/prosedur, tuliskan dengan lengkap definisi/header, spesifikasi, dan realisasi.

Contoh input dan output data (yang digaris bawah adalah masukan pengguna):

|  |  |
| --- | --- |
| **Contoh-1** | **Contoh-2** |
| Jumlah tangkapan = 5  Berat ikan (dalam gram) =  1000  2000  1500  500  100  Akumulasi poin = 300 | Jumlah tangkapan = 50  Jumlah tangkapan harus 0-20. Ulangi.  Jumlah tangkapan = 3  Berat ikan (dalam gram) =  5  5  50  Akumulasi poin = 25 |
| **Contoh-3** | **Contoh-4** |
| Jumlah tangkapan = 1  Berat ikan (dalam gram) =  7  Akumulasi poin = 0 | Jumlah tangkapan = 0  Akumulasi poin = 0 |

**Deskripsi Soal 6 dan Soal 7**

Suatu ketika, kepala sekolah **Monster University** ingin mengetahui prestasi dari para monster pada kelas yang dibuka. Data setiap monster disimpan menggunakan tipe data yang terdefinisi  sebagai berikut.

constant **NMax**: integer = 100   
   
{ NilaiMons: Nilai untuk monster dengan nomor id monster = ID, pada kelas berkode KodeKls, dengan nilai = Nilai }   
type **NilaiMons**: < ID : string,    
                   KodeKls : string,    
                   Nilai : integer[0..100] >   
   
{ Tabel berisi data nilai semua monster }   
type **ArrNilai**: < TNilai : array [1..NMax] of NilaiMons, { penampung data nilai monster }   
                  Neff : integer[0..NMax]  { Nilai efektif tabel, Neff=0 berarti tabel kosong } >    
   
{ Tabel berisi data nama-nama monster }   
type **ArrNamaMonst**: < TNM : array [1..Nmax+1] of string,    
                    Neff : integer[0..NMax] { Nilai efektif tabel, Neff=0 berarti tabel kosong } >

Untuk soal-soal berikut, tuliskan secara lengkap: **definisi/header, spesifikasi, dan body/realisasi fungsi. Tidak diperkenankan membuat dan menggunakan type/fungsi/prosedur baru.**

**Soal 6**

Buatlah fungsi **NilaiMaxKelas** yang menerima input ArrNilai (misalnya T) dan kode kelas (misalnya KodeKls), kemudian mengembalikan nilai yang merupakan nilai tertinggi untuk kelas KodeKls. Jika array kosong atau jika KodeKls tidak ditemukan di array, dihasilkan nilai -1.

**Soal 7**

Buatlah fungsi **Search**yang digunakan untuk mencari nilai sebuah string (misalnya X) pada array ArrNamaMonst (misalnya T). Fungsi menghasilkan indeks di mana X ditemukan pertama kali di T dan menghasilkan –1 jika X tidak ditemukan di T. Gunakan algoritma searching dengan sentinel di mana sentinel diletakkan di indeks terakhir.

**Deskripsi Soal 8 dan Soal 9**

Diketahui sebuah program yang digunakan untuk mengelola catatan record score dari server **game arcade online**. Setiap player dapat memainkan lebih dari satu game, dan seorang player dapat memainkan sebuah game berulang kali. Setiap kali player menyelesaikan sebuah game, score akhir yang dicapai player akan dicatat di dalam tabel score.

Contoh isi tabel score:

|  |  |  |
| --- | --- | --- |
| **ID Player** | **ID Game** | **Score** |
| 101001 | 9001 | 1234 |
| 101001 | 9001 | 107 |
| 101001 | 9002 | 10 |
| 202002 | 9001 | 0 |
| 101001 | 9001 | 567 |
| 202002 | 9001 | 5012 |
| 101001 | 9002 | 105 |
| 303003 | 9001 | 5012 |
| 202002 | 9003 | 215 |

Program ini menggunakan prosedur **ExtractScore**dan prosedur **RecordTop10**.

Prosedur **ExtractScore**digunakan untuk mengekstrak data score dengan IDPlayer tertentu dan menyimpannya ke dalam sebuah tabel score baru.

Prosedur **RecordTop10**digunakan untuk menampilkan 10 score teratas (Top 10) dalam sebuah game secara terurut mengecil. Pemain dengan score yang sama akan dikelompokkan dalam ranking yang sama.

Contoh ranking untuk ID Game 9001:

|  |  |  |
| --- | --- | --- |
| **Rank** | **ID Player** | **Score** |
| 1 | 303003 | 5012 |
| 1 | 202002 | 5012 |
| 2 | 101001 | 1234 |
| 3 | 101001 | 567 |
| 4 | 101001 | 107 |
| 5 | 202002 | 0 |

Berikut diberikan potongan kamus untuk program tersebut. Type **PlayerScore**digunakan untuk menyimpan data score seorang player per game. Type **TabScore**digunakan untuk menyimpan kumpulan data PlayerScore.

**KAMUS**  
constant NMax : integer = 1000  
type PlayerScore : < IDPlayer : integer,   
                     IDGame : integer,   
                     Score : integer >= 0 >  
type TabScore : < TS : array [1..NMax] of PlayerScore,   
               Neff : integer[0..NMax] { nilai efektif TabScore,

0 jika tabel kosong } >

Tuliskanlah implementasi dari prosedur **ExtractScore**dan prosedur **RecordTop10**sesuai dengan spesifikasi yang diberikan di bawah ini. **Header fungsi/prosedur harus ditulis ulang, namun spesifikasi tidak perlu ditulis ulang.**

**Tidak diperkenankan membuat dan menggunakan type/fungsi/prosedur baru.**

**Soal 8**

**procedure ExtractScore** (input T1 : TabScore, input ID : integer,

output T2 : TabScore)   
{  Membaca data score pemain dengan IDPlayer = ID dari T1 dan menyimpannya ke dalam T2  }   
{ I.S. : T1 terdefinisi, mungkin kosong. ID terdefinisi. T2 sembarang. }   
{ F.S. : T2 berisikan data score pemain dengan IDPlayer = ID. Mungkin kosong. }

**Soal 9**

**procedure RecordTop10** (input TS : TabScore, input IDGame : integer)   
{ Menampilkan ke layar 10 score teratas dari game dengan ID = IDGame secara terurut mengecil }   
{ Petunjuk: Anda dapat menggunakan teknik sorting dengan selection sort yang dimodifikasi }   
{ I.S. : TS terdefinisi, tidak kosong, mungkin menampung kurang dari 10 score }   
{ F.S. : 10 ranking score teratas ditampilkan secara terurut mengecil, peraih score yang sama berada dalam satu ranking yang sama. Jika hanya ada < 10 data score, ditampilkan ranking apa adanya.   
Contoh tampilan ranking sesuai contoh di atas:  
1 303003 5012  
1 202002 5012  
2 101001 1234  
3 101001 567  
4 101001 107  
5 202002 0  
}

**Deskripsi Soal 10 dan Soal 11**

Data rekening nasabah di sebuah bank disimpan di dalam sebuah file sekuensial “Daftar\_Rekening.DAT”, terurut berdasarkan nomor rekening (norek), dengan elemen rekaman sebagai berikut.

* Norek (bertipe string, untuk menyimpan nomor rekening),
* Nama (bertipe string, untuk menyimpan nama nasabah),
* Saldo (bertipe integer >= 0, untuk menyimpan saldo akhir.

Sebagai penanda akhir rekaman, digunakan Mark berupa <"","",0>

Sedangkan file Daftar\_Transaksi.DAT berisi data transaksi nasabah selama periode tertentu (misalnya dalam satu hari), terurut berdasarkan nomor rekening (norek), dengan elemen rekaman sebagai berikut.

* Norek (bertipe string, untuk menyimpan nomor rekening),
* DK (bertipe character, untuk menyimpan kode transaksi D (untuk debit) atau K (untuk kredit)),
* Jumlah (bertipe integer > 0, untuk menyimpan nominal transaksi.

Sebagai penanda akhir rekaman, digunakan Mark berupa <"",’-’,0>

|  |  |
| --- | --- |
| **Contoh isi Daftar\_Rekening.DAT** | **Contoh isi Daftar\_Transaksi.DAT** |
| <"A001", “Adi”, 110000000>  <"A002", “Boby”, 250000000>  <"A003", “Dila”, 5000000>  **<"", "", 0>** | <"A001", ‘D’, 2000000>  <"A001", ‘K’, 12000000>  <"A003", ‘K’, 1000000>  <"A003", ‘D’, 500000>  <"A003", ‘D’, 150000>  **<"", ‘-’, 0>** |

Berikut ini adalah potongan program yang digunakan oleh manajer cabang untuk mengelola data rekening nasabah di cabangnya.

**Program** Kelola\_Rekening   
{ Deskripsi: program untuk menampilkan daftar nasabah prioritas dan   
            pemutakhiran saldo rekening nasabah.   
  Input:   
     - File Daftar\_Rekening.DAT dan Daftar\_Transaksi.DAT yang sudah terisi   
     - pilihan menu pengelolaan yang akan dilakukan   
  Output:   
     - Tampilan daftar nasabah prioritas apabila dipilih menu 1   
     - File Daftar\_Rekening\_Updated.DAT apabila dipilih menu 2   
}   
**KAMUS**  
type rekRekening : <norek:string, nama:string, saldo:integer>=0>   
constant markRekening : rekRekening = <"", "", 0> {mark arsip}   
ArsipRekening,   
ArsipRekeningOut : SEQFILE of   
                    (\*) rekRkng : rekRekening   
                    (1) markRekening   
   
type rekTransaksi : <norek:string, DK:character, jumlah:integer>0>   
constant markTransaksi: rekTransaksi = <"", '-', 0> {mark arsip}   
ArsipTransaksi : SEQFILE of   
                  (\*) rekTrans : rekTransaksi   
                  (1) markTransaksi   
   
pilihan: integer[1..3]    
constant saldo\_minimal: integer = 10000   
   
function EOP\_Rekening (rek:rekRekening) -> boolean    
{ menghasilkan true apabila rek=markRekening}   
function EOP\_Transaksi (rek:rekTransaksi) -> boolean    
{ menghasilkan true apabila rek=markTransaksi}   
procedure Cetak\_Nasabah\_Prioritas    
{ I.S.: File Daftar\_Rekening.DAT berisi daftar rekening nasabah beserta saldonya.    
  F.S.: Menampilkan daftar nasabah prioritas sesuai masukan batas saldo prioritas.   
}   
procedure Update\_Saldo   
{ I.S.: File Daftar\_Rekening.DAT dan Daftar\_Transaksi.DAT sudah terisi.    
        Rekaman pada file Daftar\_Traksaksi.DAT selalu benar:   
         - norek ada pada file Daftar\_Rekening.DAT dan    
         - akumulasi transaksi tidak akan menyebabkan saldo rekening    
           kurang dari saldo minimal.   
  F.S.: Menghasilkan file Daftar\_Rekening\_Updated.DAT yang memuat    
        rekening seperti pada file Daftar\_Rekening.DAT namun dengan nilai saldo   
        yang telah dimutakhirkan berdasarkan isi dari file Daftar\_Transaksi.DAT.   
}   
   
**ALGORITMA**  
repeat   
     output("Pilihan menu:")   
     output("1. Cetak Nasabah Prioritas")   
     output("2. Pemutakhiran Data Saldo")   
     output("3. Keluar")   
     output("Masukkan pilihan menu: ")   
     input(pilihan)   
     depend on pilihan   
          pilihan = 1: Cetak\_Nasabah\_Prioritas    
          pilihan = 2: Update\_Saldo    
until pilihan = 3

**Soal 10**

Tuliskan implementasi prosedur **Cetak\_Nasabah\_Prioritas**(header/definisi prosedur harus ditulis, spesifikasi tidak perlu ditulis ulang).

Nasabah prioritas adalah nasabah yang memiliki saldo di atas suatu batas tertentu. Nilai batas dibuat tidak tetap, karena kondisi ekonomi yang belum stabil. Prosedur Cetak\_Nasabah\_Prioritas menerima masukan nilai batas prioritas (misal 100.000.000) dari pengguna kemudian memilih nasabah yang menjadi nasabah prioritas (berdasarkan data pada file Daftar\_Rekening.DAT) dan menampilkannya ke layar. Juga ditampilkan ke layar banyaknya nasabah prioritas. Apabila tidak ada nasabah prioritas, ditampilkan ke layar nilai saldo tertinggi yang dimiliki nasabah. Jika file kosong, menampilkan pesan "Arsip Daftar Rekening kosong."

Pastikan masukan batas prioritas selalu valid, yaitu sesuai dengan saldo minimal (10.000).

Gunakan model proses sekuensial dengan mark, dengan penanganan kasus kosong.

Berikut ini contoh hasil keluaran di layar untuk isi file Daftar\_Rekening seperti pada deskripsi.

Contoh 1:

Masukkan nilai batas prioritas: 100000000   
Daftar Nasabah Prioritas:   
A001, Adi, 110000000   
A002, Boby, 250000000   
Selesai membuat Daftar Nasabah Prioritas sebanyak 2 data nasabah.

Contoh 2:

Masukkan nilai batas prioritas: 300000000   
Daftar Nasabah Prioritas:   
Tidak ada nasabah prioritas.   
Saldo tertinggi: 250000000

Contoh 3:

Masukkan nilai batas prioritas: 5000   
Masukan batas prioritas harus di atas saldo minimal!

**Soal 11**

Tuliskan implementasi prosedur **Update\_Saldo**(header/definisi prosedur harus ditulis, spesifikasi tidak perlu ditulis ulang).

Prosedur Update\_Saldo akan melakukan pemutakhiran data saldo rekening yang terdapat pada file Daftar\_Rekening.DAT berdasarkan daftar transaksi yang dilakukan untuk masing-masing rekening yang tersimpan di dalam file Daftar\_Transaksi.DAT (rekaman pada kedua file terurut berdasarkan norek). Hasil pemutakhiran saldo akan disimpan di dalam file Daftar\_Rekening\_Updated.DAT.

**Nilai Saldo yang baru =** Nilai Saldo pada Daftar\_Rekening.DAT + / - nominal transaksi-transaksi terhadap rekening.

Apabila DK='D' berarti tambahan saldo sedangkan DK='K' berarti pengurangan saldo.

Proses pemutakhiran saldo tidak boleh menggunakan array mengingat tidak ada batasan jumlah rekaman yang terdapat di dalam file.

Berikut ini isi file Daftar\_Rekening\_Updated untuk masukan file Daftar\_Rekening.DAT dan Daftar\_Transaksi.DAT seperti pada deskripsi.

|  |  |
| --- | --- |
| **Daftar\_Rekening\_Updated.DAT** | **Keterangan** |
| <"A001", “Adi”, 100000000>  <"A002", “Boby”, 250000000>  <"A003", “Dila”, 4650000>  <"", "", 0> | 110.000.000 + 2.000.000 – 12.000.000 = 100.000.000  Tidak ada transaksi untuk A002, saldo tetap  5.000.000 – 1.000.000 + 500.000 + 150.000 = 4.650.000 |