Soal 1. ADT TANGGAL [Kerjakan dalam Notasi Algoritmik] (Bobot : 25%)

Diberikan definisi ADT Tanggal sebagai berikut:

```
{ Modul ADT TANGGAL }
type TANGGAL : < DD : integer[1..31], { hari }
                 MM : integer[1..12], { bulan }
                 YY : integer > 0     { tahun } >
{ *** Konstruktor : Membentuk TANGGAL dari komponen-komponennya *** }
function MakeTANGGAL (d : integer[1..31], m : integer[1..12], y : integer>0) → TANGGAL
{ Membentuk TANGGAL dengan d sebagai DD, m sebagai MM, dan y sebagai YY. }
{ *** Selektor : Mengakses komponen-komponen TANGGAL *** }
function GetHari (T : TANGGAL) → integer[1..31]
{ Mendapatkan bagian DD dari TANGGAL T }
function GetBulan (T : TANGGAL) → integer[1..12]
{ Mendapatkan bagian MM dari TANGGAL T }
function GetTahun (T : TANGGAL) → integer>0
{ Mendapatkan bagian YY dari TANGGAL T }
{ *** Primitif-Primitif Lain *** }
function Kemarin (T : TANGGAL) → TANGGAL
{ Menghasilkan TANGGAL satu hari sebelum T }
procedure BacaTanggal (output T : TANGGAL)
{ Membaca masukan TANGGAL dari keyboard dan membentuk TANGGAL dari masukan tersebut }
{ I.S.: T sembarang; F.S.: T terdefinisi dengan komponen-komponen masukan dari keyboard }
procedure TulisTanggal (input T : TANGGAL)
{ Menuliskan T ke layar }
{ I.S.: T terdefinisi; F.S.: T tertulis di layar, dengan format DD/MM/YY }
```

Kerjakan soal berikut ini. Anda tidak diperbolehkan membuat type/fungsi/prosedur tambahan.

a. Buatlah definisi, spesifikasi, dan realisasi dari fungsi **JumlahHari** yang menerima masukan dua buah TANGGAL, misalnya T1 dan T2, dan menghasilkan jumlah hari antara tanggal T1 dan T2. Asumsi T1 lebih awal atau sama dengan T2.

```
function JumlahHari (T1,T2 : TANGGAL) → integer
{ menghasilkan jumlah hari antara tanggal T1 dan T2.
  Asumsi: T1 lebih awal atau sama dengan T2 }
Kamus lokal
    T : TANGGAL
    JmlHari : integer
Algoritma
    JmlHari ← 0
    T ← T2
    while (GetTahun(T1) ≠ GetTahun(T) or GetBulan(T1) ≠ GetBulan(T) or
           GetHari(T1) ≠ GetHari(T)) do
           JmlHari ← JmlHari + 1
           T ← Kemarin(T)
    → JmlHari
```

b. Buatlah **program** yang menggunakan ADT TANGGAL dan melakukan hal-hal berikut:

- Membaca masukan dua buah TANGGAL, misalnya T1 dan T2
- Menghitung jumlah selisih hari antara T1 dan T2, misalnya N
- Menuliskan T1, T2, dan N ke layar.

Anda bebas membuat detil-detil terkait interaksi antara program dengan pengguna selain yang dinyatakan secara eksplisit pada soal.

```
program Tanggal
{ Input: dua buah tanggal
  Output: tampilan selisih antara kedua tanggal }
Kamus
    USE ADT_TANGGAL
    T1, T2 : TANGGAL
    N : integer
Algoritma:
    BacaTanggal(T1)
    BacaTanggal(T2)
    if ((GetTahun(T1) < GetTahun(T2)) or
        (GetTahun(T1) = GetTahun(T2) and GetBulan(T1) < GetBulan(T2)) or
        (GetTahun(T1) = GetTahun(T2) and GetBulan(T1) = GetBulan(T2) and
         GetHari(T1) ≤ GetHari(T2)) then   {T1 lebih awal atau sama dengan T2}
           N ← JumlahHari(T1,T2)
    else
           N ← JumlahHari(T2,T1)
    TulisTanggal(T1)
    TulisTanggal(T2)
    output(N)
```

Kerjakan pada halaman ini dan sebaliknya.

Soal 2. ADT WAKTU [Kerjakan dalam Notasi Algoritmik] (Bobot : 25%)

Diberikan definisi ADT Jam sebagai berikut:

```
{ Modul ADT JAM }
type JAM : < JJ : integer[0..23], { jam }
             MM : integer[0..59], { menit }
             DD : integer[0..59]  { detik } >
{ *** Konstruktor : Membentuk JAM dari komponen-komponennya *** }
function MakeJAM (j : integer[0..23], m : integer[0..59], d : integer[0..59]) → JAM
{ Membentuk sebuah JAM dengan j sebagai JJ, m sebagai MM, dan d sebagai DD. }
{ *** Selektor : Mengakses komponen-komponen JAM *** }
function GetJam (J : JAM) → integer[0..23]
{ Mendapatkan bagian JJ dari JAM J }
function GetMenit (J : JAM) → integer[0..59]
{ Mendapatkan bagian MM dari JAM J }
function GetDetik (J : JAM) → integer[0..59]
{ Mendapatkan bagian DD dari JAM J }
{ *** Primitif-Primitif Lain *** }
function JamToDetik (J : JAM) → integer≥0
{ Menghasilkan jumlah detik dari JAM J }
function DetikToJam (N : integer≥0) → JAM
{ Menghasilkan JAM dari jumlah detik N }
```

Kerjakan soal berikut ini. Manfaatkan sebaik-baiknya primitif- primitif dalam ADT TANGGAL dan JAM. Semua primitif yang telah didefinisikan pada kedua ADT tersebut boleh digunakan tanpa harus direalisasikan. Anda tidak diperbolehkan membuat type/fungsi/prosedur tambahan.

a. Dengan memanfaatkan ADT JAM yang diberikan di atas dan ADT TANGGAL yang diberikan pada soal 1, tuliskan definisi dan spesifikasi dari ADT WAKTU, yang terdiri dari tanggal dan jam tertentu, termasuk definisi dan spesifikasi konstruktor dan selektornya. Contoh nilai dari ADT WAKTU adalah <<25,10,2014>,<14,0,0>>, yang menyatakan tanggal 25 September 2014 jam 14:0:0.

```
{ Modul ADT WAKTU }
USE ADT_TANGGAL, ADT_JAM
type WAKTU : < T : TANGGAL, { tanggal }
               J : JAM  { jam } >
{ *** Konstruktor : Membentuk WAKTU dari komponen-komponennya *** }
function MakeWAKTU (tt : TANGGAL, jj : JAM) → WAKTU
{ Membentuk sebuah WAKTU dengan tt komponen tanggal (T) dan jj komponen jam (J) }
{ *** Selektor : Mengakses komponen-komponen JAM *** }
function GetT (W : WAKTU) → TANGGAL
{ Mendapatkan bagian tanggal dari WAKTU W (komponen T) }
function GetJ (W : WAKTU) → JAM
{ Mendapatkan bagian jam dari WAKTU W (komponen J) }
```

b. Buatlah definisi, spesifikasi, dan realisasi dari fungsi **JumlahJam** yang menerima masukan dua buah WAKTU, misalnya W1 dan W2, dan menghasilkan jumlah selisih jam antara W1 dan W2. Asumsi: W1 selalu lebih awal atau sama dengan W2.

```
function JumlahJam (W1,W2 : WAKTU) → real
{ menghasilkan jumlah selisih jam antara waktu W1 dengan W2.
  Asumsi: W1 lebih awal atau sama dengan W2 }
Kamus lokal
   DeltaHari : integer
   DeltaDetik : integer
Algoritma
   DeltaHari ← JumlahHari(GetT(W1),GetT(W2))
   DeltaDetik ← JamToDetik(GetJ(W2)) – JamToDetik(GetJ(W1))
   if (DeltaDetik < 0) then
          DeltaDetik ← 86400 + DeltaDetik; DeltaHari ← DeltaHari – 1
   → DeltaHari*24 + DeltaDetik/3600
```

Solusi berdasarkan penjelasan Bu Yani saat Kuis

```
function JumlahJam (W1,W2 : WAKTU) → JAM
{ menghasilkan JAM yang menunjukkan durasi jam antara W1 dan W2
  Asumsi: W1 lebih awal atau sama dengan W2 dan selisih waktu tidak lebih dari 24 jam
}
Kamus lokal
    DeltaHari : integer
    DeltaDetik : integer
Algoritma
    DeltaDetik ← JamToDetik(GetJ(W2)) – JamToDetik(GetJ(W1))
    if (DeltaDetik < 0) then
           DeltaDetik ← 86400 + DeltaDetik
    → DetikToJam(DeltaDetik)
```

Kerjakan pada halaman ini dan sebaliknya.

Soal 3. ADT ARRAY [Kerjakan dalam Notasi Algoritmik] (Bobot : 25%)

Lengkapi ADT Array of Integer yang diimplementasikan secara statik dan implisit dengan elemen kontigu yang tidak harus rata kiri sebagai berikut:

```
{  MODUL TABEL INTEGER DENGAN ALOKASI STATIK }
{  Berisi Definisi dan semua primitif pemrosesan tabel statik dan implisit }
{  Penempatan elemen tidak selalu rapat kiri, dan elemen-elemen selalu kontigu }
{  Kamus Umum  }
constant IdxMax : integer = 100
constant IdxUndef : integer = 0   { indeks tak terdefinisi }
constant ValUndef : integer = -999   { value tak terdefinisi }
{ Definisi elemen dan koleksi objek }
type IdxType : integer     { type indeks }
type ElType  : integer   { type elemen tabel }
type TabInt : < TI : array [1..IdxMax] of ElType > //DEFINISIKAN memori penyimpan elemen
{ Definisi, jika T : TabInt :
  Tabel kosong: ∀i(1≤i≤IdxMax) T.TI[i] = ValUndef
  Definisi elemen pertama : T.TI[GetFirstIdx(T)]
  Definisi elemen terakhir : T.TI[GetLastIdx(T)] }
{ Konstruktor: create tabel kosong  }
procedure MakeEmpty (output T : TabInt)        // LENGKAPI parameternya
{ I.S. sembarang }
{ F.S. Terbentuk tabel T kosong dengan kapasitas IdxMax elemen }
function GetFirstIdx (T : TabInt) → IdxType      // LENGKAPI parameternya
{ Mengirimkan indeks elemen pertama. Prekondisi : Tabel tidak kosong }
function GetLastIdx (T : TabInt) → IdxType       // LENGKAPI parameternya
{ Mengirimkan indeks elemen terakhir. Prekondisi : Tabel tidak kosong}
function IsFull (T : TabInt) → boolean       // LENGKAPI parameternya
{ Mengirimkan true jika penampung belum penuh, false jika sudah penuh }
```

a. Realisasikan fungsi **GetFirstIdx**, fungsi **GetLastIdx**, dan fungsi **IsFull** tanpa menulis ulang spesifikasi.

```
function GetFirstIdx (T : TabInt) → IdxType
{versi yang memanfaatkan prekondisi Tabel tidak kosong}
Kamus lokal
      i : integer;
Algoritma
      i ← 1
      while (T.TI[i] = ValUndef) do
            i ← i + 1
      {T.TI[i] <> ValUndef, elemen terdefinisi}
      → i
```

```
function GetLastIdx (T : TabInt) → IdxType
{versi yang memanfaatkan prekondisi Tabel tidak kosong}
Kamus lokal
      i : integer;
Algoritma
      i ← GetFirstIdx(T)
      while (T.TI[i] ≠ ValUndef) and (i < IdxMax) do
            i ← i + 1
      {(T.TI[i] = ValUndef) or (i = IdxMax)}
      if ( T.TI[i] = ValUndef) then
            → i - 1
      else
            → IdxMax
```

```
function IsFull (T : TabInt) → boolean
Kamus lokal

Algoritma
      → (GetLastIdx(T) – GetFirstIdx(T) + 1) = IdxMax
```

b. Realisasikan prosedur **GeserKiri** berikut ini tanpa menulis ulang spesifikasinya.

```
procedure GeserKiri (input/output T : TabInt)
{ I.S. T terdefinisi, belum penuh, tapi area kosong hanya tersedia di bagian kiri }
{ F.S. T "digeser" ke kiri sehingga seluruh elemen jadi rata kiri; area kosong jadi
berpindah ke bagian kanan }
```

```
procedure GeserKiri (input/output T : TabInt)
Kamus lokal
      i,j,last : integer
Algoritma
      i ← GetFirstIdx(T);   last ← GetLastIdx(T)
      j ← 1
      while (i <= last) do
            T.TI[j] ← T.TI[i]
            T.TI[i] ← ValUndef
            j ← j + 1
            i ← i + 1
      {i > GetLastIdx(T)}
```

c. Realisasikan prosedur **AddSortedX** berikut ini tanpa menulis ulang spesifikasinya.

```
procedure AddSortedX (input/output T : TabInt, input X : ElType)
{ I.S. T terdefinisi, terurut membesar, dan belum penuh }
{ F.S. T sudah bertambah satu elemen bernilai X dan tetap terurut membesar; area kosong
di bagian kanan berkurang 1. Jika area kosong hanya ada di bagian kiri, manfaatkan
prosedur GeserKiri untuk "menggeser" dulu seluruh elemen tabel agar raat kiri }
```

Contoh:

Masukan

T.TI : X: 9

-999	-999	2	5	7	8	10	-999	-999	-999
1	2	3	4	5	6	7	8	…	100

Keluaran:

-999	-999	2	5	7	8	**9**	**10**	-999	-999
1	2	3	4	5	6	7	8	…	100

Masukan

T.TI : X: 9

-999	-999	-999	-999	7	8	10	15	55	95
1	…	93	94	95	96	97	98	99	100

Keluaran:

7	8	**9**	10	15	55	95	-999	-999	-999
1	2	3	4	5	6	7	8	…	100

Tuliskan jawaban di balik halaman ini

Tidak diperkenankan membuat type/fungsi/prosedur lain.

```
procedure AddSortedX (input/output T : TabInt, input X : ElType)
Kamus lokal
      i,j,first : integer
      isEmpty : boolean
Algoritma
      {periksa apakah tabel kosong}
      i ← 1;   isEmpty ← T.TI[1]=ValUndef
      while i<IdxMax and isEmpty do
            i ← i + 1
            isEmpty ← T.TI[i]=ValUndef
      {i≥IdxMax or not isEmpty}

      if isEmpty then   {kosong, X dimasukkan sebagai elemen pertama}
            T.TI[1] ← X
      else
            i ← GetLastIdx(T)
            if i=IdxMax then GeserKiri(T)
```

```
first ← GetFirstIdx(T)
while (X < T.TI[i] and i>first) do
      T.TI[i+1] ← T.TI[i]
      i ← i - 1
{ X >= T.TI[i] or i=first }
if (X >= T.TI[i]) then
      T.TI[i+1] ← X
else
      T.TI[i+1] ← T.TI[i]
      T.TI[i] ← X
```

Soal 4. ADT MATRIKS [Kerjakan dalam Notasi Algoritmik] (Bobot : 25%)
Diberikan definisi dan spesifikasi ADT matriks sebagai berikut:

```
type indeks  : integer { indeks baris dan kolom }
constant BrsMin : indeks = 1
constant BrsMax : indeks = 100
constant KolMin : indeks = 1
constant KolMax : indeks = 100
type el_type : integer
type MATRIKS :
      < Mem : matrix[BrsMin..BrsMax,KolMin..KolMax] of el_type,
        NBrsEff : integer > 0, { banyaknya/ukuran baris yg terdefinisi }
        NKolEff : integer > 0  { banyaknya/ukuran kolom yg terdefinisi }
      >  { Memori matriks yang dipakai selalu di "ujung kiri atas" }

{************ DEFINISI PROTOTIPE PRIMITIF ************}
{*** Konstruktor membentuk MATRIKS ***}
procedure CreateMATRIKS (input NB,NK:integer, output M:MATRIKS)
{ Membentuk sebuah MATRIKS "kosong" berukuran NBxNK di "ujung kiri" memori }
{ I.S. NB dan NK adalah valid untuk memori matriks yang dibuat }
{ F.S. MATRIKS M terdefinisi dengan ukuran NBxNK }

{*** Selektor MATRIKS ***}
{*** Selektor Get ***}
function GetNBrsEff (M:MATRIKS) → integer { Mengirimkan banyaknya baris efektif M }
function GetNKolEff (M:MATRIKS) → integer { Mengirimkan banyaknya kolom efektif M }
function GetElmt (M:Matriks; i,j:indeks) → el_type
{ Mengirimkan elemen M dengan nomor baris i dan nomor kolom j }
function GetFirstIdxBrs (M:MATRIKS) → indeks { Mengirimkan indeks baris terkecil M }
function GetFirstIdxKol (M:MATRIKS) → indeks { Mengirimkan indeks kolom terkecil M }
function GetLastIdxBrs (M:MATRIKS) → indeks { Mengirimkan indeks baris terbesar M }
function GetLastIdxKol (M:MATRIKS) → indeks { Mengirimkan indeks kolom terbesar M }

{*** Selektor Set: Operasi mengubah nilai elemen matriks ***}
procedure SetElmt (input/output M:MATRIKS, input i,j:integer, input X:el_type)
{ I.S. M sudah terdefinisi; F.S. M[i,j] bernilai X }
```

Dengan menggunakan ADT di atas, selesaikan soal-soal di bawah ini. Spesifikasi prosedur/fungsi tidak perlu ditulis ulang. Jika perlu membuat fungsi/prosedur baru, harus ditulis definisi, spesifikasi, dan realisasinya.

a. Matriks Positif

```
function IsMatriksPositif (M : MATRIKS) → boolean
{ mengembalikan true jika seluruh elemen matriks adalah bilangan positif, > 0 }
```

```
function IsMatriksPositif (M : MATRIKS) → boolean
Kamus lokal
   i,j : indeks
   isPos : boolean
Algoritma
   isPos ← true
   i ← GetFirstIdxBrs(M)
   while (i≤GetLastIdxBrs(M) and isPos) do
         j ← GetFirstIdxKol(M)
         while (j≤GetLastIdxKol(M) and isPos) do
               if (GetElmt(M,i,j)≤0) then
                     isPos ← false
               j ← j+1
         i ← i+1
   → isPos
```

b. Flip Matriks

```
procedure FlipMatriks (input/output M : MATRIKS)
{ Procedure FlipMatriks melakukan pencerminan matriks M berdasarkan sumbu diagonal }
{ I.S.: M adalah sebuah matriks bujursangkar (jumlah kolom dan barisnya sama)}
{       dan tidak kosong}
{ F.S.: M merupakan hasil "pencerminan" dari M berdasarkan "sumbu" diagonal matriks.}
```

Elemen diagonal matriks terletak pada posisi/indeks baris yang sama dengan posisi/indeks kolom.

Ilustrasi dan contoh Flip Matriks:

Matriks A

4	5
2	6

FlipMatriks (A)

4	**2**	
5	6	

Matriks B

3	**2**	**3**	3	4	5
4	2	**1**	2	2	6
5	**6**	5	3	1	5

FlipMatriks(B)

```
procedure FlipMatriks (input/output M : MATRIKS)
Kamus lokal
   i,j : indeks
   temp : el_type
Algoritma
   i traversal [GetFirstIdxBrs(M)..GetLastIdxBrs(M)]
         j ← GetFirstIdxKol(M)
         while (j<i) do
               temp ← GetElmt(M,i,j)
               SetElmt(M,i,j,GetElmt(M,j,i))
               SetElmt(M,j,i,temp)
               j ← j+1
```

Tuliskan jawaban di balik halaman ini.