Deskripsi Persoalan

Buatlah sebuah program dalam bahasa C yang diberi nama **LISP-B** yang merupakan sebuah interpreter sederhana bahasa LISP-B. Interpreter ini menerima masukan berupa **ekspresi-SBin** yang diketikkan oleh pemrogram, melakukan interpretasi terhadap ekspresi tersebut, dan menampilkan hasilnya ke layar.
LISP-B adalah bahasa mirip LISP, namun hanya mampu menangani ekspresi biner (dengan 2 operan) dan uner (dengan satu operan). Hal ini berbeda dengan bahasa LISP yang menggunakan ekspresi-S yang menangani ekspresi dengan argumen yang dapat bervariasi. Sementera ekspresi-SBin pada LISP-B hanya menangani ekspresi biner atau uner.
Konsep-konsep dasar dalam Bahasa LISP dapat dilihat dalam **Diktat Kuliah Pemrograman Fungsional Bagian LISP** atau sumber-sumber lain, misalnya **LISP 1.5 Programmer's Manual** oleh John McCarthy dkk.

Interaksi dengan Pengguna

Ketika interpreter LISP-B dijalankan, program dikenali dengan munculnya prompt ">". Setiap ekspresi yang akan diinterpretasi, harus dituliskan setelah tanda prompt tersebut.
Seperti pada LISP biasa, setiap ekspresi dituliskan sebagai *list of symbol*s antara tanda kurung ( ). Jumlah pasangan kurung buka-kurung tutup dalam suatu ekspresi harus tepat berpasangan satu-satu. Jika tidak, maka ekspresi salah. Program versi awal yang anda buat selalu menangani ekspresi yang bebas dari kesalahan sintaks.
Akhir sebuah ekspresi ditandai dengan karakter "#". Dengan demikian, bisa terjadi ekspresi kosong, yaitu jika pengguna hanya mengetikkan "#". Ekspresi kosong tidak diinterpretasi.
Setelah satu ekspresi selesai ditulis dan benar, program akan secara otomatis menginterpretasi ekspresi dan mengeluarkan hasil. Lalu memunculkan kembali tanda prompt.
Interpreter LISP-B ditutup jika pengguna mengetikkan perintah `(exit)#`.
Contoh:

```
> (+ 1 2)#
3
> (= 1 2)#
T
> (+ (+ 1 3)
     (- 2 1))#
5
```

Atom dan List

Ekspresi-SBin dapat terdiri atas 3 jenis:

- Atom
  Suatu atom bisa numerik (integer atau real), simbol (karakter atau string), atau boolean. Boolean dinyatakan sebagai karakter "T" untuk nilai true dan "F" untuk nilai false.
  Contoh atom: `1, 2, 4, ABC, A, X, a, T, F`
- List
  Suatu list terdiri atas sekumpulan ekspresi (bisa atom, bisa list, atau campuran antara keduanya). List kosong dinyatakan sebagai ( ) atau nil. Setiap elemen list dipisahkan dengan spasi.
  Contoh list: `(1 2), nil, (1 2 a b), (abc 2 c), (a b c (1 2 3))`
- Ekspresi biasa (lihat penjelasan di bawah).

Penanda bahwa suatu ekspresi adalah atom atau list adalah diawali dengan karakter kutip tunggal " ' " (kecuali untuk boolean, tidak perlu). Ekspresi biner/uner biasa tidak perlu diawali dengan karakter kutip tunggal.
Setiap atom atau list selesai dituliskan, maka interpreter akan menuliskan ulang ekspresi tersebut.
Contoh:

```
> '(1 a 2)#
(1 a 2)
> 'a#
a
>  '( 1 2 3 (a b))#
( 1 2 3 (a b))
>  T#
T
```

Ekspresi

Program LISP-B juga harus dapat menerima masukan berupa ekspresi yang menyatakan suatu operasi dan memproses operasi tersebut serta menampilkan hasilnya.
Ekspresi dituliskan secara prefix dan dalam bentuk list. Simbol pertama dalam list tersebut adalah *operator*, sisanya adalah *operan*. Penulisan ekspresi selalu benar.

Beberapa operasi yang harus bisa dilakukan dalam LISP-B:

1. Ekspresi Aritmatika, Relasional, dan Lojik

- Ekspresi dasar aritmatika berlaku hanya terhadap atom yang bernilai numerik (integer maupun real) dan memberikan hasil numerik (integer atau real). Operasi yang mungkin: pemangkatan (exp), penjumlahan (+), pengurangan (-), perkalian (*), pembagian (div dan /), sisa pembagian (mod). Di samping itu juga ada operator minus (-) terhadap suatu nilai numerik yang memberikan hasil nilai numerik tersebut dikalikan -1 dan operator "se-per" (/) terhadap suatu nilai numerik yang memberikan hasil 1 per nilai numerik bersangkutan.
  Catatan: Pelajari konsep operasi dasar aritmatika dalam LISP terutama yang terkait dengan tipe integer dan real dan bagaimana hasil operasi terhadap kedua jenis nilai tersebut terhadap operator-operator aritmatika.
  Contoh:
  ```
  > (+ 2 (+ 3 4))#
  9
  > (* (+ 1 2) (+ (- -3 4) 4))#
  -9
  > (- -3)#
  3
  > (/ -3)#
  -0.33333
  ```
- Ekspresi dasar relasional berlaku terhadap atom yang bernilai numerik (integer dan real) atau karakter dan hasilnya adalah boolean. Operasi yang mungkin: kesamaan (=), ketidaksamaan (/=), lebih besar (>), lebih kecil (<), lebih besar atau sama dengan (>=), lebih kecil atau sama dengan (<=).
  Contoh:
  ```
  > (> 2 5)#
  F
  > (<= 4 (+ 3 2))#
  T
  ```
- Ekspresi dasar lojik berlaku terhadap ekspresi yang bernilai boolean dan hasilnya bernilai boolean. Operasi yang mungkin: negasi (not), dan (and), serta atau (or).
  Contoh:
  ```
  > (AND (> 1 2) (/= 3 4))#
  F
  > (OR T (< 1 2))#
  T
  ```

Pada interpreter LISP-B, dibatasi bahwa satu operator hanya memiliki maksimum 2 operan (ekspresi biner atau uner). Operator uner yang dapat ditangani hanyalah negasi (minus untuk nilai numerik, serta "NOT" untuk boolean).

2. Ekspresi Kondisional

- Ekspresi if-then-else, yaitu jika terdapat maksimum 2 kondisi yang komplementer dan terdapat ekspresi yang berbeda untuk tiap kondisi.
- Ekspresi cond, yaitu jika terdapat > 2 kondisi yang saling disjoin dan terdapat ekspresi yang berbeda untuk tiap kondisi.

3. Assignment

Assignment (pemberian nama untuk suatu ekspresi) ditandai dengan operator `setq`. Dalam pemrosesannya, setiap nama yang di-*assign* untuk suatu ekspresi harus disimpan supaya bisa digunakan untuk berbagai hal, termasuk untuk digunakan dalam ekspresi lain, misalnya ekspresi aritmatika atau kondisional. Ini adalah aspek prosedural untuk menyimpan hasil evaluasi ekspresi.

4. Operator Dasar terhadap List

- `car`: menghasilkan elemen pertama sebuah list, hasilnya dapat sebuah atom atau list, dengan syarat bahwa list tidak kosong (apa yang terjadi jika list kosong?).
- `cdr`: menghasilkan list tanpa elemen pertama sebuah list, hasilnya sebuah list.
- `cons`: menghasilkan sebuah list dengan cara menambahkan sebuah atom atau list ke dalam suatu list, hasilnya suatu list.
- `list`: menghasilkan sebuah list dari kumpulan atom dan list, hasilnya suatu list.
- `append`: menghasilkan sebuah list yang merupakan penggabungan 2 buah list.
- `reverse`: menghasilkan sebuah list yang terbalik urutan kemunculan elemennya.

5. Predikat Dasar

- `atom`: menghasilkan true jika simbol yang diperiksa adalah atom.
- `listp`: menghasilkan true jika simbol yang diperiksa adalah list.
- `null`: menghasilkan true jika list yang diperiksa adalah list kosong.
- `equal`: menghasilkan true jika operan-operannya bernilai sama (apa perbedaan operator kesamaan = dengan equal?).

Catatan: Lihat sumber-sumber bacaan untuk penjelasan lebih jauh mengenai konsep ekspresi dalam bahasa LISP untuk memperoleh gambaran lebih jelas.

Load dan Save File

Di samping bisa menginterpretasi suatu ekspresi yang diketikkan pengguna secara langsung, program LISP-B juga bisa menerima sebuah file teks yang berisi deretan ekspresi LISP, kemudian menginterpretasi setiap ekspresi dalam file tersebut, dan menuliskan hasilnya. Penulisan tiap ekspresi diakhiri dengan karakter "#". Setiap ekspresi yang dituliskan dalam file adalah ekspresi yang bebas dari kesalahan sintaks.
File yang dapat diterima hanyalah file teks dengan ekstensi LSPB.
Contoh file teks berisi ekspresi LISP yang diberi nama TES.LSPB:

```
(+ 2 (+ 3 4))#
(* (+ 1 2) (+ (- -3 4) 4))#
(> 2 5)#
(<= 4 (+ 3 2))#
(setq L '(1 2 3))#
(cons '1 L)#
```

Operator untuk melakukan loading file teks adalah `load`. Perintah loading file harus dituliskan dalam bentuk ekspresi juga, dengan operan nama file yang akan dibuka, dituliskan di antara tanda kutip ganda (").
Contoh eksekusi load file:

```
> (load "TES.LSPB")#
9
-9
F
T
(1 2 3)
(1 1 2 3)
```

Catatan: Semua hasil ekspresi assignment harus disimpan selayaknya ekspresi assignment biasa.

Di samping itu, program juga harus bisa melakukan penyimpanan terhadap semua ekspresi yang pernah diketikkan pengguna sebelum perintah simpan dilakukan, termasuk ekspresi yang berasal dari file eksternal yang di-load oleh pengguna. Penyimpanan ini dilakukan dalam sebuah file teks yang didefinisikan namanya oleh pengguna. Operator untuk melakukan penyimpanan file teks adalah `save`. Perintah ini mempunyai sebuah operan, yaitu nama file penyimpanan.
Contoh:

```
> (save "TESSAVE.LSPB")#
```

Perintah ini akan mengambil semua perintah sebelum save dilakukan dan menyimpannya dalam file TESSAVE.LSPB.
Baik untuk perintah load maupun save, file yang digunakan adalah file yang berada di direktori yang sama dengan interpreter LISP-B.

Evaluasi Ekspresi

Dalam LISP suatu fungsi/ekspresi bisa dievaluasi, yaitu pemeriksaan urutan eksekusi setiap ekspresi yang terkandung dalam fungsi/ekspresi tersebut.
Buatlah sebuah evaluator sederhana yang menunjukkan urutan eksekusi suatu ekspresi biner (hanya untuk operasi dasar aritmatika, relasional, dan lojik). Urutan eksekusi seperti apa, diserahkan kepada Anda dan harus Anda buat spesifikasinya. Operator untuk melakukan evaluasi adalah `eval`, diikuti operan yang berupa ekspresi yang akan dievaluasi.
Contoh:

```
> (eval (* (+ 1 2) (+ (- -3 4) 4)))#
1 (* (+ 1 2) (+ (- -3 4) 4))
2 (* 3       (+ -7       4))
3 (* 3       -3           )
4 -9
```

Bonus

Bonus dikerjakan jika dan hanya jika seluruh tugas yang wajib telah Anda selesaikan dengan baik. Beberapa pilihan bonus yang bisa Anda buat:

1. Tambahkan kemampuan untuk memroses komentar.
2. Tambahkan kemampuan untuk mendefinisikasi operator baru dengan perintah `defun`. Dalam pemrosesannya, setiap nama yang di-*assign* untuk suatu fungsi baru harus disimpan supaya bisa digunakan kembali untuk berbagai hal, termasuk saat fungsi digunakan dalam ekspresi lain, misalnya ekspresi aritmatika atau kondisional.
3. Tambahkan kemampuan untuk *error-handling* terhadap penulisan sintaks ekspresi yang salah.
4. Tambahkan kemampuan untuk mentranslasi ekspresi-SBin yang prefix menjadi ekspresi infix atau postfix (hanya mentranslasi, tidak memroses).
5. Tambahkanlah fungsionalitas help, yaitu suatu perintah yang dapat digunakan untuk memberikan daftar perintah-perintah yang tersedia dalam interpreter dan operator-operator dalam bahasa LISP, berikut penjelasannya.
6. Tambahkanlah fungsionalitas-fungsionalitas lain dalam bahasa LISP yang tidak disebutkan di atas yang bisa diadopsi ke LISP-B.

Deliverables

1. Source code, well commented, dengan pembagian file yang sesuai standar. Primitif dalam ADT hanya primitif yang dipakai dalam program.
   Tuntutan kualitas source code:
   a. *Readability* : Setiap potongan kode dapat dibaca dengan ”mudah”, a.l. diberikan komentar dengan baik.
   b. *Modularity* : Setiap unit dapat dikompilasi terpisah, jelas siapa pengembangnya, sesuai standar yang diberikan di kelas.
   c. ”Bersih” : Kode yang di-deliver hanya yang diperlukan saja.
2. Laporan yang berisi antara lain:
   - Penjelasan mengenai spesifikasi tugas yang belum spesifik dan desain struktur data internal yang digunakan untuk memecahkan tiap persoalan yang ditemukan dalam tugas. Jika mengerjakan bonus, maka berikan penjelasan mengenai spesifikasi tugas yang Anda tambahkan, berikut struktur data yang digunakan.
   - Sketsa struktur data internal yang digunakan.
   - Jika diperlukan, tambahkan penjelasan mengenai algoritma-algoritma menarik yang Anda temukan dalam tugas ini yang Anda pikir bisa menaikkan kualitas tugas Anda.
   - Data test.
   - Test script, yaitu urutan test yang "masuk akal" dan mencakup minimal semua fasilitas yang harus di-test serta sesuai spesifikasi di atas.
   - Pembagian tugas (penanggung jawab masing-masing modul program)
   - Notulen setiap pertemuan untuk klarifikasi spesifikasi yang dilakukan per kelompok.

Pengerjaan Tugas

1. Tugas ini menggunakan banyak ADT dan mesin yang pernah dan akan diajarkan di dalam mata kuliah ini, mulai dari tabel, list linier, stack, pohon, termasuk mesin karakter/mesin kata.
2. Tugas harus diserahkan per kelompok, setiap kelompok beranggotakan 5 orang. Setiap mahasiswa harus memahami spesifikasi
3. Deadline:
   a. Asistensi 1: 8 April 2013
   b. Asistensi 2: 25 April 2013
   c. Pengumpulan: 30 April 2013
4. Waktu dan tatacara penyerahan serta pemeriksaan tugas akan diumumkan di situs http://kuliah.itb.ac.id pada link kuliah IF2030/Algoritma dan Struktur Data.