Catatan Kuliah

Pemrograman Perl

Oleh :

dwi sakethi

PENGRAJIN SISTEM INFORMASI

0816-403 432

dwijim@unila.ac.id

http://dwijim.wordpress.com

tulisan niki dipun serat ngangge LaTeX

mBandar Lampung 2010-2011

Daftar Isi

Catatan Kuliah
Pemrograman Perl

dwi sakethi
dwijim@unila.ac.id,

19 Maret 2012

1 Kata Pengantar

Sebenarnya cukup berat rasanya untuk mengajar Perl ini, karena interaksi dengan Perl baru sebatas menggunakan Nikto saja ... :-) Selain itu juga, rasa-rasanya kesibukan mengajar setiap semester itu sepertinya cukup 'berlebihan'. Untuk tahun ajaran yang lalu (2007/2008) sudah terlepas dari Perl karena mata kuliah ini diasuh oleh dosen yang lain.

Namun setelah dua kali mencoba membuat program dengan Perl, salah satunya program untuk melakukan penilaian *coding*, ternyata memang enak juga menggunakan Perl. Dan terfikir bahwa nanti bakal banyak *script* yang dibuat dengan menggunakan Perl ini. Terima kasih kepada Prof. Finkel yang telah memacu munculnya ide ini.

Satu hal lain yang berbeda dengan semester-semester sebelumnya adalah penghayatan bahwa filosofi **mengajar dengan cinta** merupakan suatu keniscayaan. Selain itu entah mengapa juga, apakah ini suatu *misteri* ataukah merupakan kesadaran baru bahwa *resource sharing* sebagai lahan berbuat kebaikan berpacu dengan ritme jalan menuju perumahan abadi (kuburan).

Catatan revisi penulisan :

1. 3 Maret 2010 perubahan mencakup: perubahan materi tugas #1, penjelasan lebih terinci tentang perintah-perintah, fungsi-fungsi dasar dalam Perl, bobot penilaian yang berubah dari semester Ganjil 2008/2009.

2. Tulisan ini mulai dibuat pada masa perkuliahan semester Ganjil 2008/2009 sebagai bahan acuan perkuliahan.

Dokumen ini dapat ditelusuri di **http://dwijim.wordpress.com**. Semoga bermanfaat ...

2 Plagiat ...

Plagiat ini merupakan masalah yang sangat penting. Oleh karena itu perlu mendapat perhatian khusus. Bahan acuan tentang plagiat ini, diambil dari tulisan Prof. Finkel (lagi-lagi) yang bisa diakses di :
http://www.cs.uky.edu/∼raphael/courses/CS450/backgr.html.
Beliau menuliskan sebagai berikut :

Plagiarism

All academic work, written or otherwise, that you submit is expected to be the result of your own thought, research, or self-expression. You may certainly discuss the programs with each other, but you must not show each other your code; everyone must develop the code completely independently. It is a serious offense to allow other students to copy your work or to copy the work of other students (even if it is in a public computer file). If you borrow ideas, algorithms, wording, or code from other sources, you must acknowledge that fact or you have committed plagiarism. If you directly take more than about 4 words in a row from any source, you must indicate that you have done so, typically with a footnote or an in-line citation, using indentation or quote marks to set off the quoted text. (Some of this text is taken from the EDP202 plagiarism guide.) Offenses against this policy are punished quite strictly. I catch plagiarism practically every semester.

3 Pemrograman yang Baik

Seperti apakah kriteria pemrograman yang baik ? Salah satu bahan acuan tentang ini, diambil dari tulisan Prof. Finkel (lagi-lagi) yang bisa diakses di :
http://www.cs.uky.edu/∼raphael/checklist.html

Checklist for good programming This checklist should help you write high-quality programs. Raphael Finkel, 8/17/2005

1. Identifiers : Make sure all your identifiers are meaningful.
   (a) One-letter identifiers are almost never meaningful.
   (b) Names like flag and temp are seldom meaningful. Instead of flag, consider naming the Boolean condition it checks for, such as valueFound.
   (c) Consider multi-word identifiers, like nameIndex. Long identifiers (within reason) tend to be very readable.
2. Bare literals : Avoid numbers other than 0 and 1 and strings other than "" in your program except when you define constants.
   (a) Don't use a literal integer as an array bound.
   (b) Don't use a literal integer as a run parameter, such as a timeout or port number.
   (c) Don't use literal integers to select menu entries.
   (d) Don't use a literal integer to measure the size of a string or some data; use sizeof() and strlen() in C and C++ and .length() and .size in Java.
   (e) Don't use a literal string for a file name. You may output literal strings, though.
   (f) Don't use a literal integer to index into an array containing heterogeneous data.
   (g) Don't declare an identifier with a name denoting a literal, such as "thirty".
3. Modularization : A program is built out of interacting components.
   (a) Don't put all your code into the main() routine.

(b) In fact, don't make any routine do too much work. If it's longer than about 50 lines, it is maybe too long.

(c) If you duplicate code several times, consider whether a loop would work better, or perhaps a subroutine.

(d) If you find you are indenting very deeply, you most likely aren't using subroutines when you should.

(e) Don't reinvent library routines (unless your assignment requires it). Look in the manuals to learn about sprintf() and atoi(), for instance.

(f) Use header files in C and C++ (header files have names ending .h) to define all constants needed by multiple files and declare all subroutines exported between files. But don't put the body of subroutines in header files (with the rare exception of inline subroutines).

4. Formatting : Your program should be easy to read.

   (a) Look at http://geosoft.no/development/javastyle.html for clear suggestions on formatting and other presentation issues. This reference is specifically directed at Java, but it has value for other languages, too.

   (b) Try to restrict all your lines to 80 characters; many people view code in 80-column windows for historical reasons.

   (c) Don't use both tabs and spaces for indentation, because not all text editors treat tabs as exactly 8 spaces.

   (d) Do follow a consistent indentation pattern that reflects the program's control structure.

   (e) Don't put lots of blank lines in your program. One blank line between subroutines is enough.

   (f) Different operating systems terminate lines different ways. If you move between Win32 (which uses /r/n), Unix (which uses /n), and MacOS (which uses /r), reformat your file to use a consistent termination method.

   (g) Don't set the executable bit (Unix) on your source files.

5. Coding : You want your coding to be clear, maintainable, and efficient, in that order. Some of the rules here are very specific; others are more general.

   (a) Don't use a sequence of if statements that have no else if only one can match; use else if.

   (b) When you want to categorize text input, don't enumerate the possible first characters.

   (c) Use shift operators instead of multiplication for constructing bit patterns.

   (d) In a switch statement, always check for the default case. Likewise, in a sequence of if-then-else statements, use a final else.

   (e) All system calls can fail. Always check the return code, and use perror() to report the failure.

   (f) Booleans should always use the boolean type in Java, bool in C++, and 0/1 integers in C. Don't use characters t and f, and don't use -1 and 1.

   (g) Use loops to initialize data structures if possible.

(h) Use each variable and each field of a structure for exactly one purpose. Don't overload them unless there is an excellent reason to do so.

(i) Don't use the same identifier for both a type, a variable, and a file name, even if you change the capitalization. It's too confusing.

(j) If you are modifying data with htonl() or a similar routine before network transmission, don't modify the data in place. Build a second data structure.

(k) Try not to use global or nonlocal variables. Declare each variable in the smallest scope you can. There are legitimate uses of nonlocal variables, but make sure you really need them.

(l) Shell and Perl programs should have their # ! line as the first line of the file; otherwise, the line is just a comment.

(m) Try to avoid coding special cases. You can often use pseudo-data or other data-structure methods that allow you to fold special cases into the regular cases.

6. Compilers : Let them help you find mistakes

   (a) Always invoke compilers with all warnings enabled. For C and C++, use the -Wall flag; for Java, use -Xlint:all -deprecation.

   (b) All Perl programs should run with the -w flag and should have use strict. All Perl cgi-bin scripts should have the -T flag, too.

7. The make utility : Use it, and use it well.

   (a) A makefile should always have a "clean" recipe, which should remove all files that can be reconstructed by other recipes in the makefile, including object and executable files.

   (b) If your project has multiple source files, the makefile should generate object (.o) files as needed and link them together.

   (c) The makefile should be written so that if you run make twice in a row, the second run does no recompilation.

   (d) Every recipe should create the file specified in its target.

   (e) Every recipe should use every file specified in its prerequisite list.

   (f) Learn to use rules for targets like .c.o to avoid repetitious makefiles.

   (g) If you have just one C or C++ source file, the executable file should have the same name (without the extension .c or .cpp).

   (h) Make sure you list all .h files as prerequisites where they are needed. Consider using makedepend to generate the prerequisite list for you.

8. Documentation : It's not just just for the grader. It helps you as you write the program, too!

   (a) Add documentation as you write the program. You can always modify it as your design changes.

   (b) Include external documentation : How does one compile and run the program, and what is it meant to do ? The external documentation could be in a separate file; for small projects, it can be a comment in the single source file.

(c) Include internal documentation : What algorithms and data structures are you using? An overview can be in a separate file, but usually internal documentation is placed on the specific routines, declarations, and steps that it describes.

(d) Check your whole program and documentation for spelling mistakes. It is impolite to turn in misspelled work.

(e) Check all your documentation (and output messages) for grammar mistakes.

(f) Programs are much more readable if you put a short comment on closing braces. For instance, the brace closing a conditional can have a comment like "if value looks good". A brace closing a loop can have a comment like "for each input line". A brace closing a procedure can have a comment just naming the procedure. A brace closing a class can have a comment saying "class" and then the name of the class.

4 Ketrampilan yang Dibutuhkan dalam *Programming*

Menulis atau membuat program bukanlah pekerjaan yang mudah, akan tetapi juga tidak cukup alasan untuk mengatakannya sebagai sesuatu yang sulit. Dalam menulis program, tidak membutuhkan bakat alam atau ketrampilan bawaan, seperti misalnya menyanyi atau melukis. Dalam menyanyi atau melukis, seseorang yang tidak memiliki bakat bawaan akan terasa sulit untuk melakukannya. Namun berbeda dengan menulis program komputer. Ada 4 ketrampilan yang dibutuhkan di sini, yaitu :

1. Perhatian terhadap hal-hal yang rinci. Komputer itu barang yang bodoh tidak dapat dipercaya (*incredibly stupid*). Pemrogram tidak dapat hanya menjelaskan 3/4 bagian dari proses dan kemudian berkata, "Kamu ngerti kan dengan apa yang aku mauin ?". Jika dalam penulisan mesti ada ; maka harus ditulis. Karena variabel harus dideklarasikan, maka variabel pun harus dideklarasikan.

   Misalnya bagaimana mencari bilangan terbesar dari sekumpulan bilangan.

2. Bersikap bodoh. Sebagaimana tadi sudah disebutkan bahwa komputer bodoh tidak dapat dipercaya. Komputer hanya mengerjakan apa yang betul-betul diperintahkan kepada mereka, tidak lebih dan tidak kurang. Komputer tidak bisa diberikan mie rebus dan kemudian disuruh membaca petunjuk yang ada dibungkusnya, maka kemudian komputer akan membuat mie rebus.

3. Ingatan yang baik. Banyak hal yang harus diingat ketika membuat program. Mulai dari aturan penulisan, fungsi dan perintah yang digunakan, parameter untuk suatu fungsi yang sudah dibuat, nama-nama variabel yang digunakan dan lain sebagainya.

4. Kemampuan untuk berfikir secara abstrak dan mampu memikirkan suatu hal dalam beberapa model tingkatan. Ini merupakan kemampuan yang paling penting di dalam melakukan aktifitas pemrograman. Misalnya dalam proses melakukan pengurutan bilangan. Proses pengurutan bilangan merupakan hal yang sangat mudah dilakukan secara manual (jika datanya sedikit). Nah ... bagaimana proses yang bisa dilakukan sendiri oleh manusia ini, sekarang dialihkan dalam perintah-perintah yang bisa dipahami oleh komputer.

   Ketika Anda ditanya, "Bintang itu kecil kan ya ...?", apa jawaban Anda.

5 Aspek Penilaian Kuliah

Dalam kuliah ini aspek-aspek yang dinilai meliputi :

1. Kuis bobot 25%.
2. Tugas bobot 40%.
3. UTS bobot 10%.
4. UAS bobot 15%.
5. Praktikum bobot 10%.

Mahasiswa yang dinyatakan terkena masalah plagiat akan dikenakan hukuman berupa pemberian nilai E untuk mata kuliah ini.

6 Kriteria Penilaian Program

Kriteria yang digunakan dalam penilaian tugas, ujian atau pun kuis mencakup 3 hal yaitu :

1. *Source code*
2. Dokumentasi sistem
3. *Running program*

Kriteria paling penting untuk ketiga aspek tersebut di atas adalah tentang plagiat. Terus terang, plagiat ini sangat tidak mengenakkan bagi semuanya jika sampai terjadi. Lebih lanjut, efeknyapun bisa meluas ke sesuatu yang tidak disangka sebelumnya. Energi untuk memikirkan masalah ini sangat besar. Ibarat baterai untuk *handphone*, plagiat ini akan membuat baterai tersebut langsung *drop*. Sangat tidak menyenangkan bagi mahasiswa yang terkena masalah ini dan juga sangat tidak mengenakkan bagi dosen yang terpaksa memberikan sanksi karena masalah plagiat. Meskipun sudah diberikan berbagai macam peringatan, plagiat ini nampaknya menjadi makhluk yang tidak bisa dihilangkan tapi hanya bisa dikurangi jumlahnya. Sedari awal sudah disiapkan kalimat untuk siapa pun yang melakukan plagiat, "Selamat Anda berhasil mengelabui sesama manusia tapi tidak bisa mengelabui Sang Pencipta dan malaikat-Nya . . . ".

6.1 Bobot Penilaian *Source Code*

Bobot penilaian untuk *souce code* mengacu kepada kriteria berikut. Kriteria-kriteria ini dapat berubah sewaktu-waktu. Jika disebutkan secara khusus bahwa penilaian tidak menggunakan kriteria ini, maka penilaian akan dilakukan dengan cara lain.

```
# dwijim : ada identitas program tapi urutan salah : -1
# dwijim : akhir suatu blok tidak ditandai dengan komentar : -.5
# dwijim : bukan perl standar linux/unix : -3
# dwijim : ejaan yang salah : -1
# dwijim : identitas tidak sesuai spesifikasi : -1
# dwijim : identasi atau tabulasi yang kurang pas  : -1

# dwijim : if you directly take more than about 4 words in a
```

```
# dwijim : row from any source, HARUSNYA INI KENA PINALTI : -0

# dwijim : jumlah source code terlalu sedikit dibanding target : -50
# dwijim : keterangan tidak sesuai dengan realita : -0
# dwijim : komentar lebih spesifik lagi : -0
# dwijim : komentar lebih spesifik lagi, ukuran ada panjang, lebar, dll : -0
# dwijim : magic number, string, etc  : -.5
# dwijim : menyamakan berbeda dengan mencocokkan : -0
# dwijim : nama variabel kurang bermakna : -1
# dwijim : nama variabel panjang tidak memakai punuk unta atau tanda _ : -.5
# dwijim : nama file tidak sesuai ketentuan : -5
# dwijim : nama file yang fixed : -.5
# dwijim : nama file tidak sesuai spesifikasi : -60
# dwijim : nama file salah dan file bukan file teks : -50
# dwijim : peran error kurang spesifik : -0
# dwijim : satu baris lebih dari 80 karakter  : -1
# dwijim : sebagai nilai awal bukan nilai minimal ? : 0
# dwijim : sebaiknya tidak menggunakan huruf kapital semua : -0
# dwijim : setelah koma atau titik dua tidak perlu ada spasi : -0

# dwijim : semua program diletakkan dalam satu program utama
# dwijim : tanpa ada fungsi atau prosedur atau sub rutin
# dwijim : kecuali jika memang program tidak sampai 50 baris
# dwijim : termasuk komentar.                 :    -3

# dwijim : struktur if yang rumit : -2
# dwijim : terlalu banyak baris kosong : -.5
# dwijim : tidak ada identitas program        : -5
# dwijim : tidak menggunakan spasi : -0.5
# dwijim : tidak menggunakan use strict, use warnings, #!/usr/bin/perl : -1
# dwijim : tidak semua orang di dunia ini membaca catatan dimaksud : -0
# dwijim : tidak internal documentation sama sekali atau terlalu sedikit : -7
# dwijim : tidak diawali dengan npm, nama, email : -5
# dwijim : tidak ada penjelasan program tentang apa : -1
# dwijim : tidak perlu menggunakan ; : -0
# dwijim : tidak perlu ada spasi antar tanda / : -0
# dwijim : variabel tidak dideklarasikan : -3
-------------- komentar-komentar ---------------
# dwijim : ehm ... enak juga yang ada mau mau mengatasi masalah kita : -0
# dwijim : jadi yang tidak sebenarnya seperti apa ? : -0
# dwijim : sayang ya tinggal finishing aja ... belum tuntas ... : -0
# dwijim : kalau model kalimat, diakhiri dengan titik : -0
# dwijim : komentar yang sama persis dengan komentar sebelumnya : -2
```

6.2 Bobot Penilaian Dokumentasi Sistem

Dokumentasi sistem bukan berisi *source code* dan sama sekali tidak ada *source code*. Dokumentasi sistem berisi penjelasan-penjelasan tentang program, cara menjalankan program, alur logika program, fungsi-fungsi yang digunakan atau penjelasan lain yang dianggap perlu. Ada pun kriteria penilaiannya adalah :

```
# dwijim : ada identitas program tapi urutan salah : -1
# dwijim : ada karakter aneh yang sebenarnya tidak terpakai : -0.5
# dwijim : bagaimana dengan posisi direktori aktif ? : -0
```

```
# dwijim : bukan bahasa resmi dokumentasi sistem : -3

# dwijim : dokumentasi sistem tidak seperti ini
# dwijim : ini harusnya internal documentation

# dwijim : ejaan yang salah : -1
# dwijim : identasi atau tabulasi yang kurang pas : -1
# dwijim : identitas tidak sesuai spesifikasi : -1
# dwijim : isi kalimat mengandung pemahaman yang salah : -3
# dwijim : sebaiknya tidak menggunakan huruf kapital semua : -0

# dwijim : jumlah dokumentasi terlalu sedikit dibanding target : -50
# dwijim : lebih banyak salinan source code

# dwijim : saya bukan kami ... jadi : 3
# dwijim : kalimat kurang spesifik terhadap obyek : -0.5
# dwijim : nama file salah dan file bukan file teks : -50
# dwijim : nama-nama fungsi ditulis huruf besar : -0
# dwijim : nama file tidak sesuai : -5
# dwijim : nama fungsi salah : -1
# dwijim : salah maksud : -3
# dwijim : satu baris lebih dari 80 karakter : -1
# dwijim : sebaiknya tidak menggunakan huruf kapital semua : -0
# dwijim : setelah koma atau titik dua biasanya ada spasi : -0
# dwijim : source code tidak perlu ditampilkan lagi di sini : -0
# dwijim : terlalu banyak baris kosong : -0.5
# dwijim : tidak internal documentation sama sekali atau terlalu sedikit : -7
# dwijim : tidak diawali dengan npm, nama, email : -5
# dwijim : tidak ada penjelasan tentang alur program : -3
# dwijim : tidak ada penjelasan tentang fungsi-fungsi : -2
# dwijim : tidak ada hubungan dengan istilah memory utama : -0
# dwijim : tidak diawali dengan informasi program tentang apa : -1
# dwijim : yang penting bisa jalan pada os apa saja : -0
# hasil tugas #1
# dwijim skor script
# dwijim skor dokumentasi sistem
# dwijim nilai running program
# dwijim nilai sewaktu-waktu bisa berubah jika ditemukan kecurangan ...
# dwijim mahasiswa yang sudah mendapat nilai, maka tugas/uts tidak berpengaruh
```

7 Dokumen Tugas/Ujian

Dokumen tugas atau ujian, terdiri atas item penilaian, yaitu :

1. *Source code*, penilaian dari *source code* berdasarkan catatan-catatan yang ada pada *check list for good programming*, termasuk catatan/komentar dalam program. Bobot 25 %.

2. Dokumentasi sistem, dokumen ini mencakup antara lain : cara menjalankan program, alur program, fungsi-fungsi yang ada, dan catatan-catatan lain yang memang perlu. Bobot 25 %.

3. Hasil eksekusi program. Ukurannya adalah kesesuaian dengan spesifikasi yang ditentukan dan kebenaran hasil ekskusi. Bobot 50 %.

Semua dokumen tersebut ditulis dalam format *text based*, bukan dalam format *office*-misalnya.

8 Materi Kuliah

Materi kuliah yang direncanakan atau yang sudah dilakukan :-) :

1. Pertemuan 1 menjelaskan tentang plagiat dan konsekuensinya, kriteria pemrograman yang baik (mengambil dari Raphael Finkel), pengenalan program yang paling mudah dengan Perl. Kekuatan pemrograman Perl dan contoh melakukan penilaian *source code* dengan *script* Perl.

2. Pertemuan 2 membuat drama tentang mencari bilangan terbesar yang mengungkapkan dialog antara dua orang. Satu orang berperan sebagai orang yang ingin mencari suatu bilangan terbesar dari sekelompok bilangan yang dimilikinya. Satu lagi berperan sebagai pakar yang akan mencari bilangan tersebut. Drama ini untuk menjembatani bagaimana menterjemahkan pola fikir manusia ke dalam perintah-perintah yang bisa dimengerti oleh Perl. Pengantar Perl diingatkan tentang struktur kendali, perulangan, blok program. Kemudian pemasukan data dengan perintah <**STDIN**>. Pengenalan jenis variabel : skalar, array, dan hashing.

3. Pertemuan 3 : latihan membuat program untuk mencari bilangan terkecil kemudian dikoreksi sesama mahasiswa dengan sistem geser. Materi selanjutanya adalah membaca data dari *file*. Dikenalkan juga tentang dasar-dasar HTML. Membuat kelompok. Tugas membaca suatu *source code* kemudian menuliskan kembali isi *file* tersebut dengan memberi tambahan adanya nomor baris di awal setiap baris pada *source code* tersebut. Contoh mencetak bilangan dengan format tertentu.

   ```
   printf '<%06s>', 12;   # prints "<000012>"
   ```

   Untuk melihat *file* HTML, gunakan program *browser*.

4. Pertemuan 4 materinya mengulang dasar-dasar HTML, kemudian mengenalkan tentang *tag* <table> dan bagaimana menampilkan gambar dengan HTML. Sebagai aplikasinya adalah menghitung statistik masing-masing huruf vokal dan menampilkannya dalam bentuk grafik di dalam format HTML.

5. Pertemuan 5 materinya dasar-dasar HTML : $tag < pre >$, pilihan tambahan perintah perataaan vertikal, menjalankan *browser* dalam *script* Perl, mengulang tentang membuat prosedur.

6. Pertemuan 6 materinya dasar-dasar HTML : $tag < a\ href >$ untuk melakukan kaitan (*link*) dengan dokumen lain, pengurutan (*sorting*) *array* dengan perintah `sort` dan melakukan pengurutan sendiri. Untuk contoh diberikan pengurutan dengan metode *buble sort*.

7. Pertemuan tanggal 13 April 2009 diisi dengan ujian tengah semester yang sangat luwes karena bukan hanya *open book* tapi juga *take home*. *Goreng pisang anget-anget, UTS gue bangetsssss ...*

8. Pertemuan 7 ini sebenarnya masih ragu-ragu karena belum diketahui hasil dari evaluasi dengan Ujian Tengah Semester. Dari beberapa contoh hasil Tugas # 2 masih ada

kelompok-kelompok yang belum sampai tuntas dalam mengerjakan Tugas # 2 tentang koreksi ujian. Untuk lebih memperdalam bagi yang sudah tuntas dan untuk memberikan contoh penyelesaian sistem koreksi, maka pertemuan ini diisi dengan diskusi tentang tugas program koreksi. Bahan yang didiskusikan adalah membahas contoh penyelesaian program untuk melakukan koreksi, meliputi :

(a) Penjelasan alur program sesuai *source code* yang diberikan. Penjelasan ini dilakukan secara detail terhadap masing-masing baris dari *source code* yang diberikan. *Source code* ada di bagian akhir dari tulisan catatan kuliah ini.

(b) Membuat contoh penilaian dengan teknik *check list for good programming*. Untuk skoring masing-masing kelompok bisa membuat model sendiri-sendiri.

Kemudian masing-masing kelompok juga membuat evaluasi terhadap proses belajar mengajar. Misalnya penjelasan terlalu cepat, masalah yang dibahas terlalu sulit, dan sebagainya. Termasuk memberikan evaluasi kepada dosen pengajar. Evaluasi ini tidak akan memberi efek terhadap nilai kuliah. Jadi tidak perlu khawatir jika memberikan evaluasi negatif terhadap proses belajar mengajar yang dilakukan. Sebaliknya juga, kalaupun memberikan pujian juga tidak lantas memberikan nilai A atau B.
Tugas kelompok : masing-masing kelompok maksimal 5 orang, tugas ini tidak akan dinilai tapi menjadi syarat mutlak untuk mendapatkan nilai akhir. Artinya jika tidak mengumpulkan tugas ini, nilai akhir tidak akan keluar. Tetapi jika mengumpulkan tugas ini, skor nilai akhir tidak akan terpengaruh. Tugas ini adalah membuat program untuk menulis kalender pada suatu bulan. Titik-titik kritis yang harus diperhatikan adalah : bulan apa, tanggal satu, jumlah hari dalam bulan itu. Bentuk keluaran sama seperti kalender pada umumnya tapi ini hanya untuk satu bulan tertentu saja. Tugas ini dikumpulkan pada pertemuan kuliah 27 April 2009.

9 Pengantar Perl

Pada bagian berikut akan diberikan sedikit pengantar tentang Perl. Pembaca semoga sudah cukup memahami tentang materi-materi dasar pemograman seperti bentuk-bentuk perulangan, pengecekan kondisi, pemasukan data dan sebagainya. Perl ini merupakan materi pemrograman ke-3 yang artinya pembaca sudah mempunyai bekal yang cukup tentang pemrograman. Catatan-catatan pada bagian ini sebagian besar merupakan kutipan dari *Perl version 5.10.0 documentation*. Dokumen ini terdapat dalam paket sistem dari Perl itu sendiri.

Apakah Perl itu ? Perl merupakan singkatan dari *Practical Extraction and Report Language*. Jawaban ini bisa ditemukan pada – *Perl 5.10.0 documentation* – yang dibuat oleh Kirrily "Skud" Robert < *skud@cpan.org* >. Di dalam dokumentasi tersebut ditulis:

> *Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.*
> *The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). Its major features are that it's easy to use, supports both procedural and object-oriented (OO) programming, has powerful built-in support for text processing, and has one of the world's most impressive collections of third-party modules.*

9.1 Awal Program

Setiap program dengan bahasa Perl harus dimulai dengan perintah :

```
#!/usr/bin/perl
```

Ini untuk memberikatahukan sistem bahwa ini adalah program dengan bahasa Perl. Bentuk awalan ini, sedikit berbeda jika sistem operasi yang digunakan adalah Microsoft Windows.

9.2 Deklarasi Variabel

Berbeda dengan bahasa Pascal atau C, dalam pendeklarasian variabel Perl mempunyai model sendiri. Untuk mendeklarasikan variabel privat digunakan perintah `my`. Misalkan :

```
my $bilangan_terkecil=-9999999;
   # variabel untuk menyimpan bilangan terkecil
   # sebagai asumsi, nilai ini sebaiknya sekecil mungkin

my $tombol_y="ya";
   # memberi nilai awal masih ada data

my $bilangan_terbesar;
   # variabel untuk menyimpan bilangan terbesar

my $bilangan_yang_ada;
   # variabel untuk menyimpan bilangan yang sedang diproses

my $masih_ada_data;
   # variabel untuk menanyakan masih ada data atau tidak
```

9.3 Pemasukan Data Via *Keyboard*

Pemasukan data dalam Perl bisa dilakukan dengan perintah <STDIN>. dalam bahasa C. Contohnya adalah :

```
print "bilangannya berapa sih : ";
$bilangan_yang_ada = <STDIN>;
# memasukkan data bilangannya

print "datanya masih ada enggak [ya/tidak] : ";
$masih_ada_data = <STDIN>;
```

9.4 Pencetakan

Pencetakan dalam Perl bisa dilakukan dengan perintah `print`, sama seperti dalam bahasa C. Contohnya adalah :

```
print "bilangannya berapa sih : ";
print "Bilangan terbesar : $bilangan_terbesar \n";
print "sukses \n";
```

9.5 Perulangan

Berikut ini salah satu contoh model perulangan dalam Perl.

Perl's C-style for loop works like the corresponding while loop; that means that this:

```
for ($i = 1; $i < 10; $i++)
  {
    ...
  }
```

is the same as this:

```
$i = 1;
while ($i < 10)
{
  ...
} continue
{
  $i++;
}
```

9.6 Penyeleksian Kondisi

Berikut ini salah satu contoh model penyeleksian kondisi dalam Perl.

Secara umum sama dengan bahasa pemrograman lain, hanya *Relational Operators* yang berbeda, seperti pada catatan berikut :

1. Binary "<" returns true if the left argument is numerically less than the right argument.
2. Binary ">" returns true if the left argument is numerically greater than the right argument.
3. Binary "<=" returns true if the left argument is numerically less than or equal to the right argument.
4. Binary ">=" returns true if the left argument is numerically greater than or equal to the right argument.
5. Binary "lt" returns true if the left argument is stringwise less than the right argument.
6. Binary "gt" returns true if the left argument is stringwise greater than the right argument.
7. Binary "le" returns true if the left argument is stringwise less than or equal to the right argument.
8. Binary "ge" returns true if the left argument is stringwise greater than or equal to the right argument.

Equality Operators

1. Binary "==" returns true if the left argument is numerically equal to the right argument.
2. Binary "!=" returns true if the left argument is numerically not equal to the right argument.

10 Program Mencari Bilangan Terbesar

Berdasarkan contoh dialog dalam *mini lesson*, maka dibuatlah program untuk mencari bilangan terbesar. Untuk model-model dialog yang lain, silahkan dikembangkan sendiri programnya.

```
# ------------------------------------------------------------------
#  program untuk mencari bilangan terbesar dengan PERL
#  dibuat hari ahad 22 februari 2009 jam 07.05
#  hari minggu bikin program ... kasian dech ...
#  dengan salah satu versi, karena pasti ada versi yang lain
#  editing program :
#  - 3 maret 2010
#   ------------------------------------------------------------------

#!/usr/bin/perl
# ------------------------------------------------------------------
#   keterangan :
#   # : komentar, memberitahukan intepretes bahwa ini bukan instruksi
#   ! : menunjukkan bahwa program ini adalah program perl
#   /usr/bin/perl : menunjukkan lokasi intepreter perl
#   ------------------------------------------------------------------

use strict;
# untuk menangani masalah-masalah yang kemungkinan muncul
# dan program akan berhenti

use warnings;
# untuk memberikan peringatan ketika ada permasalahan
# sama dengan parameter -w ketika eksekusi

my $bilangan_terkecil=-9999999;
   # variabel untuk menyimpan bilangan terkecil
   # sebagai asumsi, nilai ini sebaiknya sekecil mungkin

my $tombol_y="ya";
   # memberi nilai awal masih ada data

my $bilangan_terbesar;
   # variabel untuk menyimpan bilangan terbesar

my $bilangan_yang_ada;
   # variabel untuk menyimpan bilangan yang sedang diproses

my $masih_ada_data;
   # variabel untuk menanyakan masih ada data atau tidak

$bilangan_terbesar = $bilangan_terkecil;
   #bilangan terbesar diberi harga awal sekecil mungkin

# ------------------ mulai program utama ------------------

hapus_layar();
# layar dibersihkan dulu biar enaq
```

```
$masih_ada_data="ya";
# proses akan dikerjakan selama masih ada data

#while ($masih_ada_data eq $tombol_y)
# looping ini tidak berhasil jadi dibuat seperti di bawah

while ($masih_ada_data =~ /$tombol_y/)
# looping akan dijalankan terus selama ada jawaban 'ya'
  {
    print "bilangannya berapa sih : ";
    $bilangan_yang_ada = <STDIN>;
    # memasukkan data bilangannya

    print "datanya masih ada enggak [ya/tidak] : ";
    $masih_ada_data = <STDIN>;
    # menanyakann apakah datanya masih ada atau tidak
    # ini untuk menentukan looping masih jalan terus atau stop

    #membandingkan bilangan yang baru dientri
    #dengan bilangan terbesar yang sudah ada
    #jika bilangan yang baru dimasukkan ternyata lebih besar
    #dari bilangan terbesar yang ada sekarang maka
    #bilangan terbesar yang ada sekarang diubah nilainya
    #menjadi sama dengan bilangan yang baru dientri

    if ($bilangan_yang_ada >  $bilangan_terbesar)
       { $bilangan_terbesar = $bilangan_yang_ada;}
  }

print "Bilangan terbesar : $bilangan_terbesar \n";
print "sukses \n";
# ---- akhir program utama ----

#---- sub program menghapus layar
sub hapus_layar
    {
      my @perintah_shell_linux=("clear");
      # variabel array berisi perintah-perintah yang akan
      # dieksekusi, tidak boleh variabel local
      # harus global (my)
      # karena belum tahu perintah menghapus layar di perl
      # jadi gunakan saja perintah menghapus layar di GNU Linux


      system(@perintah_shell_linux);
      # ini baris yang mengeksekusi perintah clear

    } # akhir hapus layar
```

Masih ada beberapa ide berbeda yang bisa ditemukan untuk menyelesaikan masalah ini. Oleh karena itu akan sangat baik jika Anda bisa menemukan cara Anda sendiri.

11 Berkas Data

Dalam proses pemasukan data ya

12 Spesifikasi Tugas #1 - Genap 08/09 : Nomor Baris

Tugas #1 diberikan pada pertemuan ke-3. Inti dari tugas #1 ini adalah memberi nomor baris pada suatu *source code*. Adapun rinciannya sebagai berikut :

1. Nama *file* tugas diawali dengan NPM ketua kelompok diikuti dengan **_tugas_01**. Ekstensi *file* adalah .pl untuk program dan .txt untuk dokumentasi sistem. *File* dikumpulkan dalam bentuk *softcopy* atau CD dan *hardcopy* atau dicetak di kertas.

2. *File* program dan dokumentasi sistem diawali dengan NPM, nama mahasiswa dan alamat *email* dengam format seperti berikut :

```
0717032008 : inyong ketuane    - inyong@apa.net
0717032088 : yasmin            - yasmin@olih.com
```

3. *File-file* tersebut berformat *text based*.

4. Contoh keluaran yang dihasilkan oleh program adalah :

```
00001 - use strict;
00002 - use warnings;
```

   Dan seterusnya sampai akhir program. *File* keluaran atau hasil bernama NPM ketua kelompok dan ekstensinya .out, misal 0717032008.out. Sedangkan *file* masukannya tergantung nama yang diisi ketika program dieksekusi.

5. Tugas dikumpulkan paling lambat 9 Maret 2009 jam 13.00 tanpa toleransi keterlambatan. Jika khawatir terlambat maka sebaiknya dikumpulkan diawal.

6. Belajarlah agar berhasil ...

Spesifikasi ini harus menjadi perhatian. Ketidaksesuaian dengan spesifikasi akan mengurangi atau menghilangkan nilai.
Catatan tugas #1 :

1. Tentang pasal 'take more than about 4 words ...'

2. Spesifikasi tugas : identitas dan nama file

3. Waktu pengumpulan tugas ...

13 Spesifikasi Tugas #1 - Genap 09/10 : Kamus-Kamusan

Tugas #1 diberikan pada pertemuan ke-3. Inti dari tugas #1 ini adalah menentukan apakah suatu kata termasuk kata dalam bahasa Lampung atau bukan. Atau bisa juga kata tersebut belum diketahui bahasa Lampung atau bukan jika jumlah kata-kata dalam kamus tersebut masih sedikit. Ingat bahwa ini bukan menterjemahkan kata dalam bahasa Lampung. Adapun rinciannya sebagai berikut :

1. Nama *file* tugas diawali dengan NPM ketua kelompok diikuti dengan **_tugas_01**. Ekstensi *file* adalah .pl untuk program dan .txt untuk dokumentasi sistem. *File* dikumpulkan dalam bentuk *softcopy* atau CD dan *hardcopy* atau dicetak di kertas.

2. *File* program dan dokumentasi sistem diawali dengan NPM, nama mahasiswa dan alamat *email* dengam format seperti berikut :

```
0717032008 : inyong ketuane    - inyong@apa.net
0717032088 : yasmin            - yasmin@olih.com
```

3. Nama *file* data kamus adalah kamus.dat. *File* ini berisi kata-kata dalam bahasa Lampung.

4. *File-file* tersebut berformat *text based*.

5. Contoh keluaran yang dihasilkan oleh program adalah :

```
bahasa Lampung
```

   Keluaran seperti jika kata yang dimasukkan memang ada di dalam kamus dan termasuk bahasa Lampung.

```
belum bisa ditentukan
```

   Keluaran seperti jika kata yang dimasukkan tidak ada di dalam kamus sehingga bisa jadi kata tersebut memang bukan berbahasa Lampung atau karena tidak ada di dalam kamus.

6. Tugas dikumpulkan paling lambat 21 Maret 2011 jam 13.00 tanpa toleransi keterlambatan. Jika khawatir terlambat maka sebaiknya dikumpulkan diawal.

7. Belajarlah agar berhasil ...

Spesifikasi ini harus menjadi perhatian. Ketidaksesuaian dengan spesifikasi akan mengurangi atau menghilangkan nilai.
Catatan tugas #1 :

1. Tentang pasal 'take more than about 4 words ...'

2. Spesifikasi tugas : identitas dan nama file

3. Waktu pengumpulan tugas ...

14 Dasar-Dasar HTML

Materi pemrograman di dalam Perl ini diarahkan kepada model berbasis teks karena lebih ditekankan kepada proses berfikir dalam pemrograman. Meskipun sebenarnya Perl juga bisa berbasis grafis. Untuk berbasis grafis, barangkali sekalian saja menggunakan PHP dan Apache. Untuk membantu menampilkan hasil pengolahan program, maka digunakan HTML sebagai alat bantunya. Oleh karena itu di sini perlu dikenalkan dengan dasar-dasar HTML. Tidak dibutuhkan HTML yang rumit-rumit apalagi sampai dengan Java Script.

14.1 Membuat Berkas HTML

Untuk membuat berkas HTML, bisa digunakan perangkat lunak yang bervariasi, yang penting perangkat lunak tersebut mendukung berkas berformat teks. Perangkat lunak yang bisa digunakan antara lain : vi, pico, GEdit Text Editor, NotePad, WordPad dan banyak lainnya. Perintah-perintah dalam HTML dikenal dengan sebutan *tag*. Suatu berkas HTML mesti diawali dengan *tag* $<$ *html* $>$ dan diakhiri dengan pasangannya yaitu *tag* $<$ */html* $>$. Contohnya :

```
<html>
<! awal berkas html >
<! nama berkas : contoh-html.html >

<title> Ini contoh berkas html </title>
<! membuat judul yang akan tampil di sudut kiri atas browser >

<h1> Ini grafik dari perl ke html </h1>
<! membuat tulisan ukuran agak besar (header) >

<img src=batang.jpg widht=10 height=40 >
<! menampilkan gambar >

</html>
<! akhir berkas html >
```

Jika berkas tersebut diakses menggunakan *browser*, hasilnya nampak seperti berikut :

Gambar 1: Eksekusi Berkas HTML

14.2 Tag $< title >$

Tag $< title >$ digunakan untuk menampilkan judul halaman web. Judul ini akan tampil di sebelah kiri atas halaman *browser* (biasanya). Tag ini bersifat pilihan pada suatu halaman web, jadi boleh ditulis boleh juga tidak. Contohnya :

```
<h1> Ini grafik dari perl ke html </h1>
```

14.3 Tag $< h1 >$

Tag $< h1 >$ digunakan untuk menampilkan tulisan dalam ukuran yang agak besar. Biasanya untuk judul tulisan yang menjadi penekanan tertentu. Contohnya :

```
<title> Ini contoh berkas html </title>
```

Biasanya digunakan untuk judul tulisan, judul tabel dan lain-lainnya.

14.4 Tag $< br >$

Tag $< h1 >$ digunakan untuk membuat tulisan diletakkan dalam baris berikutnya. Suatu tulisan dalam format HTML akan ditampilkan menyamping meskipun di dalam penulisannya tulis di baris yang berbeda. Perhatikan berkas HTML berikut :

```
<html>
<! awal berkas html >
<! nama berkas : tidakgantibaris.html >

1
2
3
4
5

</html>
<! akhir berkas html >
```

Hasil eksekusi dengan *browser* :

```
1 2 3 4 5
```

Sebaliknya, perhatikan berkas HTML berikut :

```
<html>
<! awal berkas html >
<! nama berkas : gantibaris.html >

1 <br>
2 <br>
3 <br>
4 <br>
5

</html>
<! akhir berkas html >
```

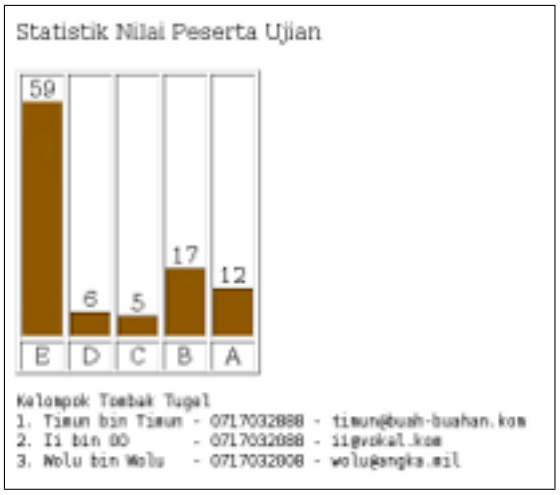
Hasil eksekusi dengan *browser* :

```
1
2
3
4
5
```

15 Spesifikasi Tugas #2 : Koreksi Ujian (Perl)

Tugas #2 diberikan pada pertemuan ke-5. Materi dalam tugas ini mencakup bahasan tentang variabel *array*, struktur kendali `if ... elsif`, penulisan *file*, pembuatan prosedur, menjalankan perintah *shell*, pemanfaatan fasilitas HTML untuk menampilkan hasil pengolahan. Fungsi-fungsi HTML yang digunakan yaitu pembuatan tabel, menampilkan gambar untuk membuat grafik, menampilkan tulisan dalam format teks asli. Ada pun proses yang dilakukan dalam program ini adalah melakukan koreksi jawaban ujian berdasarkan kunci yang sudah disediakan. Kemudian membuat skor dan menjumlah berapa orang yang mendapat nilai A, B, C, D dan E. Jumlah ini kemudian digambarkan dalam bentuk grafik dengan menggunakan format HTML. Setelah *script* dijalankan maka secara langsung akan menjalankan *browser* dan menampilkan grafik berdasarkan skor peserta ujian.

Hasil akhir dari proses tampak seperti berikut :



Gambar 2: Statistik Skor Peserta

Dalam penilaian hasil ekskusi *script* tidak ada nilai tambahan, artinya hasil akhir cukup sama dengan gambar tersebut di atas. Meskipun ada tampilan tambahan tidak akan menambah nilai.

Penjelasan rinci sekaligus merupakan spesifikasi program yang sebaiknya diikuti adalah sebagai berikut :

1. Tugas dikumpulkan hari Senin, 6 April 2009 paling lambat jam 11.00 sebelum kuliah dimulai. Jika kuliah sudah dimulai maka tugas tidak akan diterima. Seandainya ada kekhawatiran terlambat, maka sebaiknya tugas dikumpulkan tidak menjelang detik-detik terakhir. Tugas dikumpulkan langsung *face to face*.

2. Satu kelompok terdiri dari 3 orang. Jika ada kelebihan atau kekurangan anggota, maka maksimal akan ada satu kelompok yang terdiri dari 4 orang atau ada kelompok yang

terdiri dari hanya 2 orang. Jika ada lebih dari satu kelompok yang beranggotakan 4 orang atau ada yang mengerjakan sendiri, atau ada lebih dari satu kelompok yang beranggotakan 2 orang, maka tugas dari kelompok-kelompok tersebut tidak akan dinilai. Kelompok-kelompok yang punya potensi masalah adalah kelompok dengan anggota 1, 2 atau 4 orang.

3. *File* contoh berupa jawaban peserta dan kunci ujian dapat di-*download* di internet. Jika menggunakan contoh data ini, maka hasil pengolahan seperti pada gambar di atas. Jika hasil tidak sama dengan hasil tersebut, berarti *script* masih ada yang kurang tepat.

4. Penghitungan skor dilakukan dengan ketentuan yaitu jika jawaban benar maka skor +4, jika tidak menjawab atau jawaban salah maka skor -1. Kemudian berdasarkan kemungkinan nilai maksimum, maka skor dikembalikan ke nilai maksimum 100. Misalkan ada 150 soal, maka nilai maksimum adalah 600. Jadi skor akhir = skor/6. Kemungkinan akan ada nilai-nilai yang negatif tidak menjadi masalah. Model skoring di sini hanya simulasi jadi tidak memperhatikan hal-hal yang lain. Selanjutnya skor akhir digunakan untuk menentukan nilai akhir yang bisa menjadi A, B, C, D, atau E.

5. Nilai A, B, C, D atau E ditentukan dengan model seperti berikut :

   - $80 \leq skor\ akhir \leq 100$ nilai A
   - $60 \leq skor\ akhir < 80$ nilai B
   - $40 \leq skor\ akhir < 60$ nilai C
   - $20 \leq skor\ akhir < 40$ nilai D
   - $skor\ akhir < 20$ nilai E

6. Dari semua data peserta ujian, selanjutnya dicari berapa orang yang mendapatkan nilai A, B, C, D dan E. Hasil proses ini kemudian digambarkan dalam bentuk grafik dengan format HTML.

7. Untuk menjalankan program, pemakai cukup dengan menjalankan *script* Perl dan *script* Perl tersebut sudah diprogram secara langsung untuk menjalankan *browser* dan membuka *file* HTML. Tidak perlu ada tampilan atau proses macam-macam yang tidak diperlukan seperti menampilkan nama peserta, nilai atau yang lainnya.

8. Nama-nama *file* mengikuti aturan berikut :

   (a) Nama *file script* Perl adalah NPM ketua kelompok diikuti tulisan '_tugas_02.pl'. Misalnya : 0717032008_tugas_02.pl.

   (b) Nama *file* HTML adalah NPM ketua kelompok diikuti tulisan '_tugas_02.html'. Misalnya : 0717032008_tugas_02.html.

   (c) Nama *file* gambar untuk grafik batang adalah batang.jpg.

   (d) Nama *file* dokumentasi sistem adalah NPM ketua kelompok diikuti tulisan '_tugas_02.txt'. Misalnya : 0717032008_tugas_02.txt.

   (e) Nama *file* jawaban peserta adalah jawaban-peserta.dat. Sedangkan *file* kunci ujian adalah kunci.txt

   Semua *file-file* tersebut mempunyai format teks biasa kecuali tentu saja batang.jpg. Kesalahan dalam menuliskan nama *file* menjadi resiko yang harus ditanggung sendiri. Jika dalam proses pengembangan programnya menggunakan sistem operasi Windows, hendaknya memperhatikan tentang huruf besar dan huruf kecil dalam nama *file*-nya.

9. *File script* Perl dan dokumentasi sistem diawali dengan identitas mahasiswa dengan format seperti berikut :

```
inyong ketuane : 0717032008 - inyong@apa.net
yasmin         : 0717032088 - yasmin@olih.com
```

10. Kriteria penilaian tugas adalah *check list for good programming*, dokumentasi sistem dan kesesuaian hasil eksekusi *script* dengan spesifikasi dan contoh.

Spesifikasi yang sudah ditentukan di sini adalah untuk dipenuhi. Ketidaksesuaian dengan spesifikasi akan memberikan resiko sesuai dengan ketidaksesuaian tersebut. Resiko terbesar dari diabaikannya spesifikasi adalah tidak adanya nilai untuk tugas ini meskipun tugas ini sudah dikerjakan dan dikumpulkan.

Khusus tentang berbagai tindakan yang masuk dalam kategori plagiat, maka sejak tugas #2 ini tidak akan ada toleransi lagi. Hendaknya tidak ada usaha-usaha yang tidak sesuai dengan norma-norma akademis supaya segala sesuatunya berjalan dengan menyenangkan.

Semoga tugas ini memberi manfaat dan nilainya juga bagus-bagus sesuai harapan semua.

16 Materi Ujian Tengah Semester

Setelah mengerjakan Tugas #2 tentang membuat program untuk melakukan koreksi, materi ini selanjutnya dikembangkan menjadi bahan ujian tengah semester. Ujian tengah semester ini dikerjakan berkelompok dengan satu kelompok terdiri 3 orang. Ujian tengah semester ini bersifat *open book* dan *take home*. Hasil ujian dikumpulkan hari Selasa, 14 April 2009 paling lambat jam 17.00 di Gedung Matematika FMIPA Unila.

Masalah yang diusung dalam materi ujian tengah semester ini adalah tentang koreksi ujian kemudian dilakukan perankingan. Bahan-bahan *file* terdiri dari :

1. Data peserta dan jawaban disimpan dalam *file* bernama lju.txt. File ini berisi nomor ujian, nama peserta, pilihan 1 dan pilihan 2 dalam bentuk kode program studi, dan terakhir jawaban peserta.

2. Data program studi, disimpan dalam *file* ps.txt. *File* ini berisi kode kelompok program studi (1 untuk IPA dan 2 untuk IPS), kode program studi dan nama program studi.

3. Kunci jawaban untuk kelompok IPA dengan nama *file* k_ipa.txt. Kunci ini untuk pengkoreksian program studi yang masuk dalam kategori IPA.

4. Kunci jawaban untuk kelompok IPS dengan nama *file* k_ips.txt. Kunci ini untuk pengkoreksian program studi yang masuk dalam kategori IPS.

Proses koreksi dilakukan dengan mencocokkan jawaban dengan kunci jawaban. Jika pilihan 1 dari peserta tersebut adalah kelompok IPA maka digunakan kunci kelompok IPA. Demikian juga dengan kelompok IPS. Oleh karena itu ada kunci IPA dan kunci IPS. Hasil koreksi ini kemudian diranking menurut pilihan #1 atau pilihan #2. Format hasil ranking bebas, yang penting minimal mencakup nomor peserta, nama peserta, nilai dan keterangan pilihan lain. Misalnya seperti berikut :

```
Daftar Nominasi Ujian
Program Studi : Ilmu Komputer
```

```
1 xxxxx dwi sakethi                    2 Matematika
2 xxxxx ponari                         - -
3 xxxxx monggo kerso                   1 Fisika
```

Dari informasi tersebut terlihat bahwa dwi sakethi pilihan 1-nya adalah Ilmu Komputer dan pilihan 2 adalah Matematika. Sedangkan ponari pilihan 1 adalah Ilmu Komputer dan tidak memiliki pilihan 2. Monggo kerso mempunyai pilihan 1 Fisika dan pilihan 2 Ilmu Komputer. Tampilan informasi dimulai dari menu pilihan jurusan yang diinginkan kemudian dibuat *link* yang akan mengarahkan ke daftar peserta ujian masing-masing program studi. Semua hasil ditampilkan dalam format HTML yang akan langsung dijalankan setelah *script* Perl untuk melakukan koreksi dan perankingan dijalankan.

Khusus untuk ujian tengah semester ini, penilaian hanya melihat hasil *running*. *Source code* dan dokumentasi sistem tidak akan dinilai seperti halnya pada tugas # 1 dan # 2.

16.1 Kriteria Penilaian

Panduan penilaian hasil UTS :

1. Mendekati benar nilai 95.
2. Running tapi error nilai 80.
3. Program error ketika dijalankan 60.
4. Tidak ada hasil 60.
5. Terlalu sedikit 50.

17 Materi Ujian Tengah Semester Genap 2009/2010

Setelah mengerjakan Tugas #1 tentang program kamus-kamusan, materi ini selanjutnya dikembangkan menjadi bahan ujian tengah semester. Ujian tengah semester ini dikerjakan sendiri-sendiri. Ujian tengah semester ini bersifat *close book* dan dikerjakan di kampus. Hasil ujian dikumpulkan langsung hari Rabu, 24 Maret 2010 setelah selesai ujian di Gedung Matematika FMIPA Unila.

Masalah yang diusung dalam materi ujian tengah semester ini adalah tentang membuat kamus untuk menterjemahkan kata dari Bahasa Lampung ke Bahasa Indonesia atau sebaliknya. Ada pun ketentuannya sebagai berikut:

1. Data kamus disimpan dalam *text file* bernama lampung.dat. File ini berisi kata dalam bahasa Lampung kemudian diikuti tanda ":" (titik dua) dan diikuti dengan kata dalam bahasa Indonesia. Contoh :

   ```
   bangek becong : sangat enak
   lawang : pintu
   ```

2. Program dapat menterjemahkan kata dari bahasa Indonesia ke bahasa Lampung atau sebaliknya dan memberikan pesan "kata tersebut tidak ditemukan dalam kamus" jika ternyata kata tersebut belum ada di dalam data kamus.

3. Jika kata dicari ada di dalam kamus maka program hanya menampilkan arti dari kata tersebut tanpa embel-embel lain.

17.1 Kriteria Penilaian

Khusus dalam ujian ini, kriteria penilaian hanya berdasarkan logika pemrograman yang digunakan dan prediksi hasil eksekusi program. Dengan demikian untuk sementara catatan-catatan di dalam *check list for good programming* tidak digunakan.

17.2 Pelaksanaan UTS

UTS dilaksanakan dengan tua gelombang. Gelombang pertama adalah peserta dengan nomor urut daftar hadir ganjil (ingat nomor urut daftar hadir, bukan nomor NPM). Waktu yang digunakan 60 menit. Selanjutnya peserta dengan nomor daftar hadir genap dan waktu ujian juga 60 menit.

18 Program Membaca File Data

```
# ------------------------------------------------------
# source code ini diambil dari contoh tutorial Perl
# kemudian diberi tambahan :
# 1. komentar
# 2. variabel nama file
# 3. hapus layar dengan perintah shell
# contoh ini sebagai pengantar tugas membuat program
# untuk menambahkan nomor baris pada suatu source code
# bandar lampung, 14 februari 2009
# jam 01.12 ... baru sadar kalau ini adalah ulang tahun
# pernikahan yang ke 16 tahun
# ------------------------------------------------------

#!/usr/bin/perl
# ---------------------------------------------------------------------
#   keterangan :
#   # : komentar, memberitahukan intepretes bahwa ini bukan instruksi
#   ! : menunjukkan bahwa program ini adalah program perl
#   /usr/bin/perl : menunjukkan lokasi intepreter perl
#   ---------------------------------------------------------------------

use strict;
# untuk menangani masalah-masalah yang kemungkinan muncul

use warnings;
# untuk memberikan peringatan ketika ada permasalahan

my $file;
# variabel untuk menyimpan nama file yang akan dibaca

my @perintah_shell;
# variabel untuk menyimpan perintah shell yang akan dieksekusi

hapus_layar();
print "nama file yang akan dibaca :";
$file = <STDIN>;
hapus_layar();

open (INPUT, $file) || die "tidak bisa membuka file $file $!";
# membuka file yang akan diakses

     while (<INPUT>) # baca data sampai akhir file
           {
# looping selama belum akhir file eof()

               chomp;
# remove a trailing record separator from a string

               print "$_ \n";
# mencetak isi file yang dibaca, nama variabelnya $_ ... :-(
```

```
          } # akhir <input>

close(INPUT) || die "tidak dapat menutup $file $!";
# tutup file yang sudah selesai diakses

sub hapus_layar()
 {
   @perintah_shell = ("clear");
   # perintah shell yang akan dieksekusi adalah clear : menghapus layar

   system(@perintah_shell);
   # jalankan perintah menghapus layar $
 }
```

19 Program Menulis File Teks

Contoh program untuk mencetak ke *file* teks. Contoh di sini lebih kepada penulisan ke *file*-nya.

```
# ---------------------------------------------------------------------
#  program untuk mencetak tulisan ke file  dengan PERL
#  dibuat hari senin 23 februari 2009 jam 18.16
#  sambil menunggu buka puasa ...
#  file yang ditulis berformat html
#   ---------------------------------------------------------------------

#!/usr/bin/perl
# ---------------------------------------------------------------------
#   keterangan :
#   # : komentar, memberitahukan intepretes bahwa ini bukan instruksi
#   ! : menunjukkan bahwa program ini adalah program perl
#   /usr/bin/perl : menunjukkan lokasi intepreter perl
#   ---------------------------------------------------------------------

use strict;
# untuk menangani masalah-masalah yang kemungkinan muncul
# dan program akan berhenti

use warnings;
# untuk memberikan peringatan ketika ada permasalahan
# sama dengan parameter -w ketika eksekusi

my $file_hasil=">coba.html";
# menentukan nama file hasil

open(OUTFILE,$file_hasil);

print OUTFILE "<html>\n";
print OUTFILE "<title>Selamat datang </title>\n";
print OUTFILE "coba aja\n";
print OUTFILE "</html>\n";
close (OUTFILE);
```

```
// akhir sampai di sini
```

20 Contoh Program untuk Melakukan Penilaian

```
#!/usr/bin/perl

#use strict;
# jika menggunakan ini, maka fh00 jadi error
# makanya bingung juga tapi untuk sementara yang
# penting programnya jalan jadi diabaikan saja
# untuk menjalankan program ini perintahnya :
# perl ./grading.pl sample-grading.pl
# - grading.pl adalah nama program ini
# - sample-grading.pl adalah file yang sudah
#   dinilai dan akan dihitung skornya berapa

use warnings;

my $file;
my $filename;
my $input;
my $tanda_penilaian;
my $jumlah_nilai;
my $tanda_grader;
my $posisi_titik_dua;
my $panjang_baris;
my $nilai;
my $posisi_nilai;

    # process argument list of files along with any includes

    sub process
      {
        my($filename, $input) = @_;
        $input++;      # this is a string increment
        unless (open($input, $filename))
            {
              print STDERR "Can't open $filename: $!\n";
              return;
            }

        local $_;
        $jumlah_nilai = 100;
while (<$input>) { # note use of indirection
    if (/^#include "(.*)"/) {
process($1, $input);
next;

    }
    #... # whatever
```

```
            $tanda_penilaian = "grader :";
            // tanda penilaian ini harus disesuaikan
            // dengan tanda atau ciri dari penilai
            // contohnya :
            // grader : variabel i kurang bermakna : -1

            if ($_ =~ /$tanda_penilaian/)
               {
                 print "$_";
                 $posisi_titik_dua = index($_,"-");
                 $panjang_baris = length($_);
                 $posisi_nilai = $panjang_baris-$posisi_titik_dua;
                 $posisi_nilai = $posisi_titik_dua-1;
                 $nilai = substr($_,$posisi_nilai,3);
                 $jumlah_nilai = $jumlah_nilai + $nilai;
                 print "$jumlah_nilai \n";
               }
        }
    }

    foreach $file (@ARGV)
    {
        my $kirim_parameter = "fh00";
        process($file, $kirim_parameter);
    }
```

21 Program Koreksi Ujian

Contoh program ini untuk bahan diskusi kelompok pada kuliah 20 April 2009. Program ini dicetak ke kertas dan dibawa sebagai bahan diskusi kelompok.

```
00001 - # ------------------------------------------------
00002 - #   program untuk mengkoreksi ujian multiple choice
00003 - #   dibuat di waktu kuliah dan diterusin di rumah
00004 - #   program ini  akan membaca  kunci jawaban
00005 - #   kemudian  mencocokkannya dengan jawaban peserta
00006 - #   skor yang didapat kemudian dikelompokkan ke dalam
00007 - #   kategori A, B, C, D dan E.
00008 - #   hasil jumlah pengelompokkan ditampilkan dengan
00009 - #   menggunakan html
00010 - #   ------------------------------------------------
00011 -
00012 - #!/usr/bin/perl
00013 - # untuk memberi tahu komputer bahwa in program perl
00014 -
00015 - use warnings;
00016 -
00017 - use strict;
00018 -
00019 - my $file_kunci_jawaban="kunci.txt";
00020 - #misalnya nama file kunci jawaban adalah kunci.txt
00021 -
00022 - my $semua_kunci;
```

```
00023 - # untuk menyimpan hasil pembacaan satu baris kunci
00024 - #
00025 - my $jumlah_soal;
00026 - # untuk menyimpan nilai jumlah soal
00027 - #
00028 - my $posisi_nomor;
00029 - # variabel looping dari nomor 0 sampai jumlah soal
00030 -
00031 - my $posisi_asli;
00032 - # untuk menyimpan nomor yang tampil di layar
00033 - # karena array dimulai dari 0 tapi nomor  dalam keseharian
00034 - # dimulai dari 1
00035 -
00036 - my $jawaban;
00037 -    $jawaban = "";
00038 -
00039 - my @kunci_ke;
00040 - # ingat deklarasi variabel array memakai tanda @ bukan $
00041 -
00042 - my @perintah_hapus_layar;
00043 - # variabel array untuk menyimpan perintah shell yang akan diberikan
00044 -
00045 - my $jumlah_a;
00046 -    $jumlah_a = 0;
00047 - # untuk menyimpan berapa orang yang mendapatkan nilai a
00048 -
00049 - my $jumlah_b;
00050 -    $jumlah_b = 0;
00051 - # untuk menyimpan berapa orang yang mendapatkan nilai b
00052 -
00053 - my $jumlah_c;
00054 -    $jumlah_c = 0;
00055 - # untuk menyimpan berapa orang yang mendapatkan nilai c
00056 -
00057 - my $jumlah_d;
00058 -    $jumlah_d = 0;
00059 - # untuk menyimpan berapa orang yang mendapatkan nilai d
00060 -
00061 - my $jumlah_e;
00062 -    $jumlah_e = 0;
00063 - # untuk menyimpan berapa orang yang mendapatkan nilai e
00064 -
00065 - #-----------------------------------------------------------------
00066 - membersihkan_layar();
00067 - # membersihkan layar sebelum melakukan proses ...
00068 -
00069 - membaca_kunci();
00070 - # membaca kunci jawaban dan menyimpannya ke variabel memory
00071 -
00072 - mencocokan_jawaban_dengan_kunci();
00073 - # menghitung skor peserta dan membuat jumlah a, b, c, d, dan e
00074 -
00075 - menulis_hasil_ke_html();
```

```
00076 -
00077 - # akhir program utama
00078 - #-----------------------------------------------------------------
00079 -
00080 - #ini prosedur untuk membaca file kunci
00081 - sub membaca_kunci
00082 - {
00083 -   open(INPUT,$file_kunci_jawaban);
00084 -   #prose membaca data kunci
00085 -
00086 -   while (<INPUT>)
00087 -        {
00088 - #        print "kuncinya $_\n";
00089 -         $semua_kunci = $_;
00090 -         $jumlah_soal = length($semua_kunci)-1;
00091 -        } #akhir while membaca file kunci
00092 -   $jumlah_soal = $jumlah_soal-1;
00093 - #  print "jumlah soal : $jumlah_soal \n";
00094 -
00095 -   # proses memotong-motong kunci menjadi bagian per nomor
00096 -   # data ini nantinya disimpan ke variabel array ...
00097 -   for ($posisi_nomor=0;$posisi_nomor<$jumlah_soal;$posisi_nomor++)
00098 -       {
00099 -         $posisi_asli = $posisi_nomor + 1;
00100 -         # karena array  dimulai dari 0 tapi nomor mulai dari 1
00101 -
00102 -         $jawaban = substr($semua_kunci,$posisi_nomor,1);
00103 -         # memotong-motong kunci menjadi kunci per nomor
00104 -
00105 - #        print "jawaban no : $posisi_asli : $jawaban \n";
00106 -         $kunci_ke[$posisi_nomor] = $jawaban;
00107 -         # menyimpan tiap kunci ke variabel kunci ke nomor sekian ...
00108 -
00109 -       } # akhir looping dari nomor 0 sampai jumlah soal
00110 -
00111 -   # sekarang isi kunci jawaban disimpan
00112 -   # dalam variabel $kunci_ke[0], $kunci_ke[1], dst ...
00113 -
00114 -   print "jumlah soal : $jumlah_soal \n";
00115 - }
00116 - # akhir prosedur membaca file kunci
00117 -
00118 - #ini prosedur untuk membaca file kunci
00119 - sub mencocokan_jawaban_dengan_kunci()
00120 - {
00121 - # file jawaban peserta mempunyai format sebagai berikut :
00122 - # kolom 1-5    : nomor peserta
00123 - # kolom 6-13   : kosong dan tidak digunakan
00124 - # kolom 14-31  : nama peserta
00125 - # kolom 31     : jenis kelamin tidak diolah dalam kasus ini
00126 - # kolom 32-181 : jawaban peserta
00127 - # file jawaban ini berformat file teks biasa
00128 -
```

```
00129 - my $file_jawaban;
00130 - # file jawaban peserta
00131 - $file_jawaban = "jawaban-peserta.dat";
00132 -
00133 - my $jawaban_ke;
00134 - # variabel untuk menyimpan jawaban setiap nomor dari peserta
00135 - $jawaban_ke = " ";
00136 -
00137 - my $baris_jawaban;
00138 - # satu baris data jawaban peserta yang nantinya akan dipotong-potong
00139 -
00140 - my $no_peserta;
00141 - # variabel untuk menyimpan no peserta ujian
00142 -
00143 - my $nama_peserta;
00144 - # variabel untuk menyimpan nama peserta ujian
00145 -
00146 - my $jenis_kelamin;
00147 - # variabel untuk menyimpan jenis kelamin
00148 -
00149 - my $skor;
00150 - # variabel untuk menyimpan skor yang diperoleh peserta
00151 -
00152 -   open(INPUT_jawaban,$file_jawaban);
00153 -   while (<INPUT_jawaban>)
00154 -        {
00155 -          $baris_jawaban = $_;
00156 -          $no_peserta = substr($baris_jawaban,0,5);
00157 -          # mengambil no peserta dari variabel baris jawaban
00158 -
00159 -          $jenis_kelamin = substr($baris_jawaban,30,1);
00160 -          # mengambil jenis kelamin dari variabel baris jawaban
00161 -
00162 -          $nama_peserta = substr($baris_jawaban,12,17);
00163 -          # mengambil nama peserta dari variabel baris jawaban
00164 -
00165 - #         print "$no_peserta - $nama_peserta - $jenis_kelamin";
00166 -
00167 -          $skor = 0;
00168 -          # nilai diberi harga awal 0
00169 -          # akan bertambah 4 jika jawaban benar
00170 -          # dan akan berkurang 1 jika jawaban salah atau tidak menjawab
00171 -
00172 -          for ($posisi_nomor=0;$posisi_nomor<$jumlah_soal;$posisi_nomor++)
00173 - #         for ($posisi_nomor=0;$posisi_nomor<149;$posisi_nomor++)
00174 -             {
00175 -               $posisi_asli = $posisi_nomor + 31;
00176 -               # jawaban pertama ada pada kolom 31
00177 -
00178 -               $jawaban_ke = " ";
00179 -               $jawaban_ke = substr($baris_jawaban,$posisi_asli,1);
00180 -               # mengambil jawaban peserta dari variabel baris jawaban
00181 -               # per huruf
```

```
00182 - #               print "$kunci_ke[$posisi_nomor] : $jawaban_ke : $posisi_nomor : $
00183 -
00184 -               if ( $jawaban_ke eq $kunci_ke[$posisi_nomor])
00185 -                  {
00186 -                    $skor = $skor + 4;
00187 -                  }
00188 -               else
00189 -                  {
00190 -                    $skor = $skor - 1;
00191 -                  }
00192 -
00193 -             } # akhir looping dari posisi nomor 0 sampai jumlah soal
00194 - #         print "$no_peserta - $nama_peserta - $jenis_kelamin ";
00195 -          $skor = $skor / 6;
00196 -          # nilai maksimum 600 karena ada 150  soal
00197 -          # supaya kembali ke 100, maka nilai dibagi 6
00198 -
00199 - #         print "nilai : $skor\n";
00200 -
00201 -          # buat pengelompokkn dari skor yang sudah diperoleh
00202 -          if ($skor >= 80)
00203 -             {
00204 -               $jumlah_a = $jumlah_a + 1 ;
00205 -             }
00206 -          elsif ($skor >= 60)
00207 -             {
00208 -               $jumlah_b = $jumlah_b + 1 ;
00209 -             }
00210 -          elsif ($skor >= 40)
00211 -             {
00212 -               $jumlah_c = $jumlah_c + 1 ;
00213 -             }
00214 -          elsif ($skor >= 20)
00215 -             {
00216 -               $jumlah_d = $jumlah_d + 1 ;
00217 -             }
00218 -          else
00219 -             {
00220 -               $jumlah_e = $jumlah_e + 1 ;
00221 -             }
00222 -
00223 -        } # akhir looping membaca data jawaban peserta
00224 -   close(INPUT_jawaban);
00225 - }
00226 -
00227 - # prosedur untuk membersihkan layar dengan perintah clear
00228 - sub membersihkan_layar
00229 - {
00230 -   @perintah_hapus_layar=("clear");
00231 -   system(@perintah_hapus_layar);
00232 - }
00233 - # akhir prosedur menghapus layar
00234 -
```

```
00235 - # mencetak hasil rekap skor ke file html
00236 - sub menulis_hasil_ke_html
00237 - {
00238 -  my $file_html=">grafik.html";
00239 -
00240 -  open(OUTFILE,$file_html);
00241 -
00242 -  print "Jumlah a : $jumlah_a \n";
00243 -  print "Jumlah b : $jumlah_b \n";
00244 -  print "Jumlah c : $jumlah_c \n";
00245 -  print "Jumlah d : $jumlah_d \n";
00246 -  print "Jumlah e : $jumlah_e \n";
00247 -
00248 -  print OUTFILE "<html>";
00249 -  print OUTFILE "Statistik Nilai Peserta Ujian <br><br>";
00250 -  print OUTFILE "<table border=1>";
00251 -  print OUTFILE "<tr>";
00252 -
00253 -  my $tinggi;
00254 -
00255 -  print OUTFILE "<td valign=bottom align=center>";
00256 -  print OUTFILE "$jumlah_e<br>";
00257 -  $tinggi = $jumlah_e*3;
00258 -  print OUTFILE "<img src=kuning.jpg width=30 height=$tinggi>";
00259 -  print OUTFILE "</td>";
00260 -
00261 -  print OUTFILE "<td valign=bottom align=center>";
00262 -  print OUTFILE "$jumlah_d <br>";
00263 -  $tinggi = $jumlah_d*3;
00264 -  print OUTFILE "<img src=kuning.jpg width=30 height=$tinggi>";
00265 -  print OUTFILE "</td>";
00266 -
00267 -  print OUTFILE "<td valign=bottom align=center>";
00268 -  print OUTFILE "$jumlah_c <br>";
00269 -  $tinggi = $jumlah_c*3;
00270 -  print OUTFILE "<img src=kuning.jpg width=30 height=$tinggi>";
00271 -  print OUTFILE "</td>";
00272 -
00273 -  print OUTFILE "<td valign=bottom align=center>";
00274 -  print OUTFILE "$jumlah_b <br>";
00275 -  $tinggi = $jumlah_b*3;
00276 -  print OUTFILE "<img src=kuning.jpg width=30 height=$tinggi>";
00277 -  print OUTFILE "</td>";
00278 -
00279 -  print OUTFILE "<td valign=bottom align=center>";
00280 -  print OUTFILE "$jumlah_a <br>";
00281 -  $tinggi = $jumlah_a*3;
00282 -  print OUTFILE "<img src=kuning.jpg width=30 height=$tinggi>";
00283 -  print OUTFILE "</td>";
00284 -
00285 -  print OUTFILE "</tr>";
00286 -
00287 -  print OUTFILE "<tr>";
```

```
00288 -  print OUTFILE "<td valign=bottom align=center>";
00289 -  print OUTFILE "E";
00290 -  print OUTFILE "</td>";
00291 -
00292 -  print OUTFILE "<td valign=bottom align=center>";
00293 -  print OUTFILE "D";
00294 -  print OUTFILE "</td>";
00295 -
00296 -  print OUTFILE "<td valign=bottom align=center>";
00297 -  print OUTFILE "C";
00298 -  print OUTFILE "</td>";
00299 -
00300 -  print OUTFILE "<td valign=bottom align=center>";
00301 -  print OUTFILE "B";
00302 -  print OUTFILE "</td>";
00303 -
00304 -  print OUTFILE "<td valign=bottom align=center>";
00305 -  print OUTFILE "A";
00306 -  print OUTFILE "</td>";
00307 -
00308 -  print OUTFILE "</tr>";
00309 -
00310 -  print OUTFILE "</table>";
00311 -
00312 -  print OUTFILE "<pre>";
00313 -  print OUTFILE "Kelompok Tombak Tugel<br>";
00314 -  print OUTFILE '1. Timun bin Timun - 0717032888 - timun@buah-buahan.kom<br>';
00315 -  print OUTFILE '2. Ii bin OO       - 0717032088 - ii@vokal.kom<br>';
00316 -  print OUTFILE '3. Wolu bin Wolu   - 0717032008 - wolu@angka.mil<br>';
00317 -  print OUTFILE "</pre>";
00318 -
00319 -  print OUTFILE "</html>";
00320 -
00321 -  close(OUTFILE);
00322 -
00323 -   @perintah_hapus_layar=("firefox grafik.html");
00324 -   system(@perintah_hapus_layar);
00325 - }
00326 -
00327 -
```

22 Materi Praktikum

22.1 Praktikum #1

Praktikum pertama ini targetnya adalah pengenalan bahasa Perl. Materinya adalah membuat program untuk menghapus layar dan kemudian menampilkan nama dan npm ke layar. Penekanan di sini adalah pengenalan *editor* vi, cara menjalankan *script* yang dibuat dengan Perl.

Pada akhir praktikum pertama, sebaiknya mahasiswa diingatkan untuk menyiapkan

materi praktikum kedua di rumah. Dengan demikian diharapkan ketika sudah di depan komputer, mahasiswa tidak bingung lagi apa yang akan dikerjakan.

22.2 Praktikum #2

Masalah pada praktikum kedua adalah mencari bilangan terkecil kedua dari sekumpulan bilangan. Misalnya ada sekumpulan bilangan : 11, 3, 10, 17, 5, 9, 10. Maka bilangan terkecil kedua adalah 5.