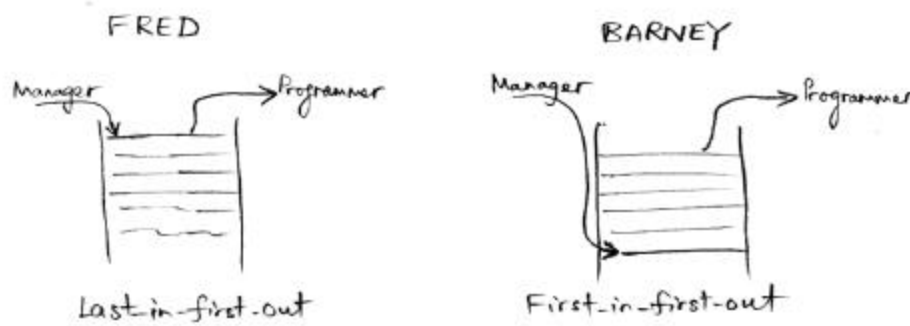


Contoh Soalan 1:

Bayangkan satu keadaan di tempat kerja yang melibatkan dua orang pengurus bernama Fred dan Barney dan dua orang 'programmers'. Programmer pertama menerima arahan daripada Fred manakala programmer yang kedua menerima arahan daripada Barney. Kedua-dua pasukan ini mengerjakan projek-projek yang berasingan. Terdapat dua 'bekas fail masuk' yang berasingan dari mana kedua-dua programmers itu menerima arahan kerja mereka masing-masing. Bila setiap satu programmer menyempurnakan satu tugas, dia mengambil arahan kerja yang seterusnya dari bahagian atas bekas fail masuk yang dimilikinya. Biasanya, Fred meletakkan arahan kerja yang seterusnya di bahagian atas bekas fail masuk bagi programmernya. Sebaliknya, Barney sentiasa meletakkan arahan kerjanya di bahagian bawah bekas fail masuk kepunyaan programmer yang menurut perintah beliau (sila lihat gambarajah di bawah).



Tulis satu program simulasi komputer samada di dalam bahasa pengaturcaraan C, C++ atau Java untuk mendapatkan purata dan sisihan piawai anggaran jumlah masa penyempurnaan sebuah tugas bagi setiap tertib pemprosesan yang dihuraikan di atas. Jumlah masa penghabisan yang dimaksudkan disini merujuk kepada jumlah masa sebuah tugas pengaturcaraan menunggu gilirannya dan masa yang diperlukan oleh programmer yang berkenaan untuk menghabiskan tugas itu. 'Bekas fail' yang tersebut di atas mestilah diwakilli sebagai satu 'linked list' bagi kedua-dua kes.

Bagi tujuan latihan ini, katakanlah masa di antara ketibaan tugas-tugas bersebelahan adalah bertaburan diantara 4 dan 7 hari dengan sama rata bagi kedua-dua kes. Begitu juga, anggaplah bahawa masa yang diperlukan oleh setiap satu programmer untuk menyempurnakan setiap tugas pengaturcaraan bertaburan secara sama rata di antara 3.9 dan 6.9 hari. Dapatkan anggaran-anggaran yang dikehendaki berdasarkan kepada 200,000 tugas yang dijanakan secara rawak.

Bagi mereka yang tidak pernah mempelajari Statistiks, sisihan piawai bagi jumlah masa penyempurnaan sebuah tugas adalah diberikan oleh pengungkapan yang berikut:

$$S = \sqrt{\frac{\sum_{i=1}^n x_i^2}{n} - \bar{X}^2}$$

dimana x_i , \bar{X} dan n masing-masing mewakili jumlah masa penyempurnaan tugas nombor i , masa purata untuk menyempurnakan sebuah tugas dan bilangan tugas.

Contoh Soalan 2:

Pada suatu masa yang akan datang, sebuah syarikat penyelidikan bernama Gammaswitch telah mencipta sebuah peranti pengkomputeran baru dengan kelajuan ultra-tinggi yang beroperasi secara pensuisan foton-foton bertenaga tinggi selain daripada pensuisan cas elektrik dengan peranti-peranti semikonduktor. Komputer Gammaswitch adalah beribu-ribu kali lebih pantas daripada peranti-peranti semikonduktor yang terpantas.

Seorang penceroboh (*hacker*) komputer muda yang berkeusahawanan bernama Bob Yates telah mencipta satu-satunya sistem pengoperasian yang berdaya maju secara komersial untuk komputer Gammaswitch. Ia merupakan gergasi sistem yang senibinanya tercabar dan hampir tidak berfungsi yang dipanggil Braindead.

Sebuah perkhidmatan maklumat Internet bernama Regurgital mengoperasikan perkhidmatan maklumat 'free-to-wire'. Perkhidmatan Regurgital telah menjadi sangat popular sehinggakan pelayan mesejnya telah terlampau dibebani dan menyebabkan kesan penangguhan yang teruk dan kesulitan kepada pelanggan-pelanggan Regurgital.

Regurgital kini ingin menggantikan komputer yang melebihi beban tersebut dengan salah satu mesin baru yang berkelajuan ultra-tinggi. Tugas anda adalah untuk memprogram komputer Gammaswitch dalam bahasa pengaturcaraan C atau C++, tetapi dalam semua kes, di bawah sistem pengoperasian Braindead yang menggerunkan, untuk menyalin mesej-mesej maklumat dari fail cakera pada komputer Gammaswitch dan menghantarkan mesej-mesej tersebut kepada pemacu (*driver*) HTTP Braindead.

Program anda mestilah berupaya untuk mengendalikan sehingga 20 muat turun (*downloads*) secara serentak.

Jika anda menulis dalam bahasa pengaturcaraan C atau C++, Braindead membekalkan fungsi-fungsi sokongan pelayan HTTP yang berikut:

GetRequest	memulakan penerimaan suatu permintaan HTTP.
PutData	memulakan transmisi sebuah blok teks output.
RequestDone	memberitahu Braindead bahawa permintaan telah dikhidmatkan dengan sempurna.
Yield	membebaskan kawalan kepada Braindead.

GetRequest mempunyai pengisytiharan berikut:

```
int GetRequest ();
```

GetRequest memulakan satu integer yang dikenali sebagai pengecam saluran (*channel identifier*). Pengecam saluran ini digunakan untuk menghubungkan operasi-operasi rangkaian yang seterusnya dengan jempukan untuk menerima sambungan HTTP yang dihasilkan oleh panggilan GetRequest. Panggilan-panggilan kepada GetRequest boleh

dilakukan satu demi satu. Setiap saluran yang terbuka akan diberikan satu nombor saluran yang berbeza oleh GetRequest.

Fungsi GetRequest hanya mendirikan suatu jemputan untuk menerima satu sambungan HTTP. Ia sebenarnya tidak menghasilkan sambungan kepada komputer pelanggan. Apabila, dalam keadaan sewajarnya, sebuah komputer pelanggan cuba untuk menyambung kepada pelayan HTTP, Braindead akan memanggil fungsi aplikasi yang dipanggil 'RequestReady' untuk memberitahu aplikasi tersebut bahawa satu sambungan telah dibuat. Aplikasi tersebut mestilah mengisytiharkan satu fungsi yang dipanggil RequestReady. Jika tidak, ia akan gagal untuk menyambung. RequestReady mestilah mempunyai pengisytiharan yang berikut:

```
void RequestReady (int chanId,  
                  const char *fileName);
```

Argumen 'chanId' akan mengandungi pengecam saluran yang dipulangkan oleh panggilan kepada GetRequest yang menghasilkan saluran komunikasi tersebut.

Argumen 'fileName' adalah penunding (*pointer*) kepada nama fail yang perlu dipulangkan kepada komputer pelanggan. Memori yang dialamatkan oleh fileName hanya akan mengandungi nama fail itu semasa pelaksanaan RequestReady. Apabila RequestReady mengembalikan kawalannya, Braindead boleh menulis semula teks nama fail tersebut.

Aplikasi tersebut boleh menggunakan fungsi-fungsi C biasa, fopen, fread dan fclose untuk membaca fail daripada simpanan besar-besaran yang berpaut dengan komputer Gammaswitch.

Untuk memulangkan kandungan fail kepada pelayan yang memintanya, aplikasi tersebut mesti memanggil fungsi PutData Braindead. PutData mempunyai pengisytiharan berikut:

```
void PutData (int chanId  
             const char *data, int len);
```

Argumen 'chanId' adalah pengecam saluran yang dipulangkan oleh panggilan asalnya kepada GetRequest.

Argumen 'data' adalah penunding kepada blok data yang mengandungi data yang perlu dituliskan.

Argumen 'len' adalah panjang keseluruhan blok data tersebut.

Panggilan kepada PutData hanya memulakan pemindahan data. PutData biasanya akan membuat pemulangan sebelum tamatnya pemindahan data. Apabila transmisi data telah selesai, Braindead memanggil fungsi aplikasi 'PutDone' untuk memberitahu aplikasi bahawa proses transmisi telah selesai.

Pengisytiharan untuk PutDone adalah:

```
void PutDone (int chanId);
```

Argumen 'chanId' adalah pengecam saluran yang dipulangkan oleh panggilan asalnya kepada GetRequest.

Braindead tidak menyalin kandungan blok data yang dihantar kepada PutData ke dalam penimbal dalaman. Sebaliknya, ia menyimpan satu penunding kepada blok data tersebut di dalam segmen memori aplikasi. Sebagai akibatnya, adalah mustahak aplikasi tersebut tidak mengubah blok data tersebut ataupun membebaskan memori blok data dengan memulangkan daripada suatu fungsi yang mana ia telah diisytiharkan sebagai pembolehkan setempat sebelum Braindead memanggil fungsi PutDone.

Apabila fail tersebut telah siap dihantar dengan panggilan berulang kepada PutData, aplikasi tersebut mesti memanggil fungsi RequestDone untuk memberitahu Braindead bahawa permintaan tersebut telah dikhidmatkan dengan sempurna. RequestDone mempunyai pengisytiharan berikut:

```
void RequestDone (int chanId);
```

Argumen 'chanId' adalah pengecam saluran yang asalnya dipulangkan oleh GetRequest.

RequestDone menutup saluran komunikasi yang dihasilkan oleh GetRequest dan membebaskan pengecam saluran. Untuk menerima permintaan yang lain, aplikasi tersebut harus membuat panggilan yang selanjutnya kepada GetRequest. Sebaik sahaja RequestDone telah membebaskan satu pengecam saluran, adalah mungkin bahawa panggilan yang seterusnya kepada GetRequest akan memulangkan pengecam saluran yang sama.

Untuk membolehkan Braindead mempunyai peluang untuk melakukan pemprosesan dalaman sendirinya serta memanggil fungsi-fungsi GetReady dan PutDone, aplikasi tersebut mesti membebaskan kawalan kepada Braindead dengan memanggil fungsi Yield. Pengisytiharan bagi Yield adalah:

```
void Yield ();
```

Apabila aplikasi memanggil fungsi Yield, Braindead mengimbas peranti-peranti yang melakukan input dan output fizikal dan memeriksa sama ada sesuatu proses input atau output itu telah selesai atau tidak. Sekiranya process sedemikian telah selesai, Braindead memanggil sama ada GetReady atau PutDone, bergantung kepada sifat permintaan yang asalnya. Jika sesuatu proses input atau output itu belum selesai, Yield akan membuat pemulangan selepas Braindead telah memanggil sama ada GetReady atau PutDone. Jika tiada proses input atau output yang belum selesai, Yield akan membuat pemulangan dengan serta-merta.

Pengisytiharan-pengisytiharan bagi fungsi-fungsi dan titik kemasukan panggilan semula aplikasi bagi Braindead adalah terkandung dalam fail 'braindead.h'. Sebuah simulasi bagi Braindead untuk Unix atau Windows adalah terkandung dalam fail 'braindead.c'.