第 12 回の課題について

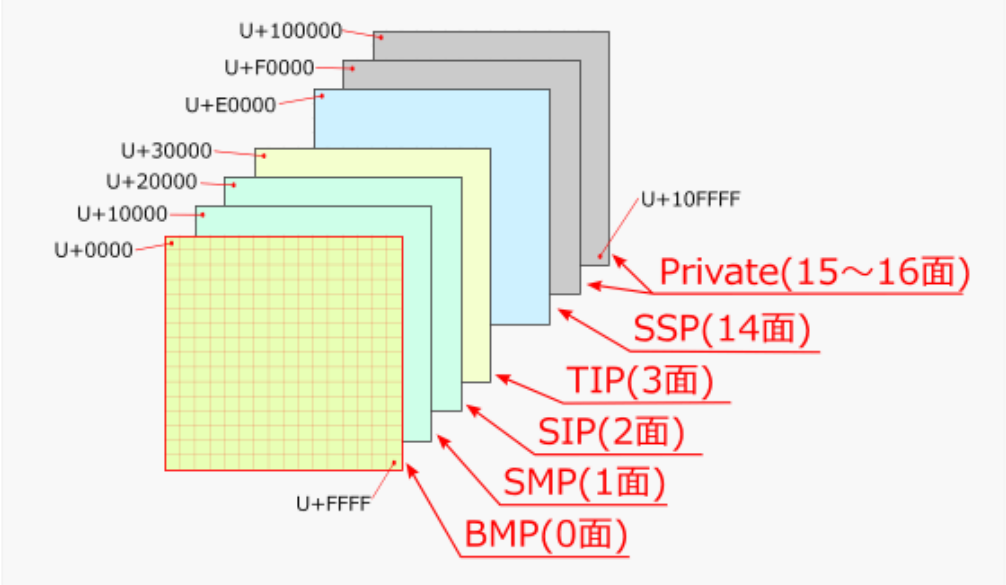
（1）文字コードをすべて（英数字、漢字とも）2 バイトで統一して扱うことの利点と問題点を述べよ

利点としては、すべての文字が 2 バイトという固定の長さで表現され（OS やすべてのアプリケーションではそれだけを使用する）と、処理が単純になることがあげられます。比較の対象をマルチバイト表現の文字コードとすると、文字によって長さが異なる場合は、たとえば 1 バイトで表現されているものなのか 2 バイトか、または 3 バイトなのか、などを区別するためにそれを識別するコードを付加することが必要になり、そのための処理が必要ですが、2 バイト固定ではそのような処理は必要ありません。

問題点は、格納に要する記憶域が増加することです。たとえば、（1 バイトで表現できる）ラテン文字が大多数を占める文書を格納する場合にすべての文字を 2 バイトで扱うと、要する記憶域が増加するということになります。

1 バイトで表現する場合に比べて表現できる文字の個数が非常に多くできるということも利点ではあります。ただし、この問題の解答としては、2 バイトでの統一表現とマルチバイト表現との比較を主にする方が良いのではと思います。

これに関連して、2 バイト固定では多いといえども表現できない文字が出る、ということを問題点としてあげていただいた方もいらっしゃいました。そのとおりで 2 バイト（16 ビット）では 65536 コードポイントしかないので、すべての文字を区別することは不可能です。（Unicode はもともと、すべての言語の文字を一つの文字コード体系で表現することを目的に 16 ビットの空間で開発されたものですが、これは不可能であり Unicode をベースとして 21 ビット空間をもつ ISO/IEC 10646 が制定されました）



2 バイト（に限らず複数バイト）で表現されてメモリ上に配置されたり順に送信されたりすると、プロセッサによって、下位のアドレス（や先に送られたもの）に上位バイトが来る場合と下位バイトが来る場合があり、バイト列として見た並びの順序（「バイトオーダ」や「エンディアン」と呼ばれます）が異なることがあります。2 バイト（に限らず複数バイト）表現の場合は、このエンディアンの問題があります。

（2）末尾が 10～59 のいずれかである文字列（たとえば、data25、file23、files40 など）をワイルドカードとメタキャラクタを使って表せ

末尾が 10～59 のいずれかである文字列ということで、まず
　*「末尾の 2 桁」
となり、末尾の 2 桁を考えると 1 桁目が 1～5 ですので [12345]、2 桁目が [0123456789] で、この 2 つを結合したもので

　[12345][0123456789] となります。（[1-5][0-9]でも結構です）
　　└ ~ でなく - です

1 文字以上あって、末尾 2 桁ということで ?*[1-5][0-9] も正解です。

*{[1-5][0-9]} や *{[1-5],[0-9]} のような解答がありましたが、{s_1,s_2,s_3,･･･} は s_1 か s_2 か s_3 か ・・・ これら複数個から一つずつということです。
{　} 内に要素が s_1 1 つだけの場合、s_1 とマッチするものを生成しても良いように思いますが、「,」がないと機能しないようです（Linux の bash の場合）
{[1-5],[0-9]} などのような場合は、[1-5]とマッチするもの、[0-9] とマッチするものと 2 つ生成されます（たとえば、ls コマンドで指定すると、コマンドが 2 つ出力されます）

なお、[α-β] の α や β は 1 文字ですので、数値としての範囲とは違って [10-59] とはできません。文字コード表での連続する並びで範囲が指定できます。

『[1 - 5]』ように、読み易さのために、記号の間に空白を入れてあるように見える解答もありますが、コマンドで指定する場合は空白もメタキャラクタ（引数の区切り）として意味を持ちますので、注意が必要です。

正規表現を使用して解答された方もいらっしゃいましたが、この問題では講義で説明したシェルのワイルドカード（ここでの「*」は正規表現のものとは違います）とメタキャラクタの範囲で解答いただければと思います。