1.

sample#1.c を解析し、ASCII コード(0x00～0x7f)の各範囲(Scope)を判断するプログラムを作成せよ。範囲例）数字：figures、英大文字：capital letter、英小文字：small letter，非表示文字：Not Printable character、その他：misc.

ソースコード：

```
/*
  program    :  Specify Scope
  Student-ID :  075739A
  Author     :  TSUHAKO Masaki
  Comment    :  範囲の特定をする
/*

#include <stdio.h>

#define FALSE 0
#define TRUE  !FALSE

int main(){
  int  value,c, count;
  char line[128];

  count = 0;

  while(TRUE){
    count++;
    if(count > 5) break;

    printf("Enter a HexValue ===> ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%x", &c);

    printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c);
    if(0x20 <= c && c <= 0x7e)
```

```
      printf("-%%c(%c)\n",c);
    else
      printf("-Not Printable character\n");

    if      (0x20 <= c && c <= 0x2f){
      puts("====> Scope_A\n");
    }else if(0x30 <= c && c <= 0x39){
      puts("====> Scope_B\n");
    }else if(0x3a <= c && c <= 0x40){
      puts("====> Scope_C\n");
    }else if(0x41 <= c && c <= 0x5a){
      puts("====> Scope_D\n");
    }else{
      puts("====> Scope_E\n16);
    }
  }

  return(0);
}
```

考察:

まずはこのプログラムの分析。

i. define とは何か。これはコンパイル直前にプログラムを置き換え、その置き換えられたものがコンパイルされる。プリプロセッサと呼ばれるものである。FALSE を 0。TRUE を!FALSE としている。

ii.16 進数を入力してください、と表示され、入力すると、ScopeA,B,C,D,E のいずれかがが出力される。

iii.Scope の範囲は次のようなものである。

A の場合：16 進数で 0x20 以上 0x2f 以下の範囲を入力

B の場合：16 進数で 0x30 以上 0x39 以下の範囲を入力

C の場合：16 進数で 0x3a 以上 0x40 以下の範囲を入力

D の場合：16 進数で 0x41 以上 0x5a 以下の範囲を入力

E の場合：16 進数で上以外のを入力（0x7f までの範囲）

iiii. while(TRUE){

count++;

if(count > 5) break;を解析

while 文は繰り返しを意味する。上のプログラムを文章にしてみると、『while が真ならば繰り返す。繰り返すことは、count が 5 より小さいのならば 1 を足していく。この条件にあわなくなったら終了（break）』

v.&&とは何か。

名前は論理 AND。似ているもので&（ビット単位 AND）がある。論理 AND は演算数が両方とも真であれば結果が 1 になる。つまり、論理 AND は演算数全体を 1 つとして演算を行う。ビット単位 AND は 1 つ 1 つのビットごとに演算を行う。

vi.else と if

考察の iiii を参考にすると、(0x20 <= c && c <= 0x2f)は、『c が 0x20 以上、0x2f 以下ならば、』という意味。そして puts にかかれた範囲が出力される。もし、上の範囲外の数字（ここは 16 進数）なのならどうするのか。その問題を解決するのが else なのである。else は英語で『ほかの』という意味がある。プログラミングでも同じ意味があり、if の条件にあわなかった範囲を else で決めておくと else の範囲で決められた範囲が出力される。

以上より、

『if と else if の条件文を変更し、printf の出力文字を変更すれば ASCII コードの各範囲を判断するプログラムになる』と判断。

C 実践プログラミング第 3 版、p407 より

表示	16 進数
数字：figures	30～39
英大文字：capital letter	41～5a
英小文字：small letter	61～7a
非表示文字：Not printable character	00～1f,7f
記号,又は演算子：misc	20～2f,3a～40,5b～60, 7b～7e

ということが判明。

上の表を元にプログラムを作り直す。

実験：

```
#include <stdio.h>

#define FALSE 0
#define TRUE  !FALSE

int main(){
  int  value,c, count;
  char line[128];

  count = 0;

  while(TRUE){
    count++;
    if(count > 6) break;

    printf("Please enter a HexValue ===> ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%x", &c);

    printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c);
    if(0x20 <= c && c <= 0x7e)
      printf("-%%c(%c)\n",c);
    else
      printf("-Not Printable character\n");

    if      (0x20 <= c && c <= 0x2f){
      puts("====> misc\n");
    }else if(0x30 <= c && c <= 0x39){
      puts("====> figures\n");
    }else if(0x3a <= c && c <= 0x40){
      puts("====> misc\n");
    }else if(0x41 <= c && c <= 0x5a){
```

```
      puts("====> capital letter\n");
    }else if(0x5b <= c && c <= 0x60){
      puts("====> misc\n");
    }else if(0x61 <= c && c <= 0x7a){
      puts("====> small letter\n");
    }else if(0x7b <= c && c <= 0x7e){
      puts("====> misc\n");
    }else{
      puts("====> Not printable character\n");
    }
  }

  return(0);
}
```

出力結果：

```
Please enter a HexValue ===> 30
Colum=01:%d( 48)-%x(30)-%c(0)
====> figures

Please enter a HexValue ===> 41
Colum=02:%d( 65)-%x(41)-%c(A)
====> capital letter

Please enter a HexValue ===> 61
Colum=03:%d( 97)-%x(61)-%c(a)
====> small letter

Please enter a HexValue ===> 61
Colum=04:%d( 97)-%x(61)-%c(a)
====> small letter

Please enter a HexValue ===> 1f
```

```
Colum=05:%d( 31)-%x(1f)-Not Printable character
====> Not printable character

Please enter a HexValue ===> 20
Colum=06:%d( 32)-%x(20)-%c( )
====> misc
```

以上の結果より ASCII コードの各範囲を判断するプログラムになっている。

2.

sample#2.c のプログラムの動作を考察せよ。

ソースコード

```
#include <stdio.h>

#define FALSE 0
#define TRUE  !FALSE

int main(){
  int  count;

  count = 0;
  while(TRUE){
    count++;
    if(count > 5) break;
    printf("While-Count=%2d\n",count);
  }

  for(count=1; count<= 5; count++){
    printf("for  -Count=%2d\n",count);
  }

  return(0);
```

```
}
```

出力結果：

```
While-Count= 1
While-Count= 2
While-Count= 3
While-Count= 4
While-Count= 5
for  -Count= 1
for  -Count= 2
for  -Count= 3
for  -Count= 4
for  -Count= 5
```

考察：

このプログラムの中には while 文と for 文が使われている。

この二つを分析する。

sample2 の中の while 文

while(TRUE){
count++;
if(count > 5) break;
printf("While-Count=%2d\n",count);
}
	条件が TRUE ならば、count に＋1 をしていく。Count の値が 5 より大きくなったら（FALSE になったら）終了。
	『While-Count=count の値（10 進数)』が出力。

sample2 の中の for 文

for(count=1; count<= 5; count++){
printf("for -Count=%2d\n",count);
}
	初期文は count=1,条件は count<=5,
	反復文は count++より、count の値、1 が 5 と等しい又は大きくなるまで count に+1 をしていく。『for -Count=count の値（10 進数)』が出力。

分析した文章を見てみると同じ内容（出力文字が少し違うだけ）という事が分かる。その証拠に出力結果が同じである。

While-Count= 1	for -Count= 1
While-Count= 2	for -Count= 2
While-Count= 3	for -Count= 3
While-Count= 4	for -Count= 4
While-Count= 5	for -Count= 5

よって、while 文と for 文を使う 2 通りの方法があることが分かった。

3.

sample#3.c を解析し、表示可能な文字による ASCII コード表を作成せよ。

ソースコード

```
#include <stdio.h>

int main(){
  int c;

  for(c = 0x20; c<=0x40; c++){
    if((c % 4) == 0) printf("\n");
    printf("%%x(%x)-%%c(%c) | ",c,c);
  }
  printf("\n");

  return(0);
}
```

出力結果：

```
%x(20)-%c( ) | %x(21)-%c(!) | %x(22)-%c(") | %x(23)-%c(#) |
%x(24)-%c($) | %x(25)-%c(%) | %x(26)-%c(&) | %x(27)-%c(') |
%x(28)-%c(() | %x(29)-%c()) | %x(2a)-%c(*) | %x(2b)-%c(+) |
%x(2c)-%c(,) | %x(2d)-%c(-) | %x(2e)-%c(.) | %x(2f)-%c(/) |
%x(30)-%c(0) | %x(31)-%c(1) | %x(32)-%c(2) | %x(33)-%c(3) |
%x(34)-%c(4) | %x(35)-%c(5) | %x(36)-%c(6) | %x(37)-%c(7) |
%x(38)-%c(8) | %x(39)-%c(9) | %x(3a)-%c(:) | %x(3b)-%c(;) |
%x(3c)-%c(<) | %x(3d)-%c(=) | %x(3e)-%c(>) | %x(3f)-%c(?) |
```

%x(40)-%c(@) |

考察：

ifor 文を見ると、『最初は 0x20（初期値）。0x40 以下ならば（条件）、c に 1 を足していく（反復文）』ということがわかった。

ii 出力結果をみると、0x20~0x40 までを ASCII コードで出力している。（考察 i と一致）

以上より

表示可能な ASCII コードは、0x20～0x7e までなので、条件の部分を 0x7e 以下に設定すればいい。

文字型は printf の%c 変換を使用するのでそのままで可能。

プログラム作成

```
#include <stdio.h>

int main(){
  int c;

  for(c = 0x20; c<=0x7e; c++){
    if((c % 4) == 0) printf("\n");
    printf("%%x(%x)-%%c(%c) | ",c,c);
  }
  printf("\n");

  return(0);
}
```

出力結果

```
%x(20)-%c( ) | %x(21)-%c(!) | %x(22)-%c(") | %x(23)-%c(#) |
%x(24)-%c($) | %x(25)-%c(%) | %x(26)-%c(&) | %x(27)-%c(') |
%x(28)-%c(() | %x(29)-%c()) | %x(2a)-%c(*) | %x(2b)-%c(+) |
%x(2c)-%c(,) | %x(2d)-%c(-) | %x(2e)-%c(.) | %x(2f)-%c(/) |
%x(30)-%c(0) | %x(31)-%c(1) | %x(32)-%c(2) | %x(33)-%c(3) |
%x(34)-%c(4) | %x(35)-%c(5) | %x(36)-%c(6) | %x(37)-%c(7) |
```

%x(38)-%c(8) | %x(39)-%c(9) | %x(3a)-%c(:) | %x(3b)-%c(;) |

%x(3c)-%c(<) | %x(3d)-%c(=) | %x(3e)-%c(>) | %x(3f)-%c(?) |

%x(40)-%c(@) | %x(41)-%c(A) | %x(42)-%c(B) | %x(43)-%c(C) |

%x(44)-%c(D) | %x(45)-%c(E) | %x(46)-%c(F) | %x(47)-%c(G) |

%x(48)-%c(H) | %x(49)-%c(I) | %x(4a)-%c(J) | %x(4b)-%c(K) |

%x(4c)-%c(L) | %x(4d)-%c(M) | %x(4e)-%c(N) | %x(4f)-%c(O) |

%x(50)-%c(P) | %x(51)-%c(Q) | %x(52)-%c(R) | %x(53)-%c(S) |

%x(54)-%c(T) | %x(55)-%c(U) | %x(56)-%c(V) | %x(57)-%c(W) |

%x(58)-%c(X) | %x(59)-%c(Y) | %x(5a)-%c(Z) | %x(5b)-%c([] |

%x(5c)-%c(\) | %x(5d)-%c()) | %x(5e)-%c(^) | %x(5f)-%c(_) |

%x(60)-%c(`) | %x(61)-%c(a) | %x(62)-%c(b) | %x(63)-%c(c) |

%x(64)-%c(d) | %x(65)-%c(e) | %x(66)-%c(f) | %x(67)-%c(g) |

%x(68)-%c(h) | %x(69)-%c(i) | %x(6a)-%c(j) | %x(6b)-%c(k) |

%x(6c)-%c(l) | %x(6d)-%c(m) | %x(6e)-%c(n) | %x(6f)-%c(o) |

%x(70)-%c(p) | %x(71)-%c(q) | %x(72)-%c(r) | %x(73)-%c(s) |

%x(74)-%c(t) | %x(75)-%c(u) | %x(76)-%c(v) | %x(77)-%c(w) |

%x(78)-%c(x) | %x(79)-%c(y) | %x(7a)-%c(z) | %x(7b)-%c({} |

%x(7c)-%c(|) | %x(7d)-%c()) | %x(7e)-%c(~) |

以上より表示可能な文字による ASCII コード表の完成。

4.

文字（文字列では無い）の演算について考察せよ。例）('a'-'A')?、('f'-'a')?

i.文字の演算を筆算で行いどのような法則があるのか調べる。

ii.簡単な ASCII コードの覚え方がないか考える。

考察 4.1:小文字から大文字を引くとどうなるか、また小文字と大文字を足すとどうなるか。

文字	10 進数	16 進数	8 進数	文字	10 進数	16 進数	8 進数
a	97	61	141	A	65	41	101
h	104	68	150	H	72	48	110
q	113	71	161	Q	81	51	121
z	122	7a	172	Z	90	5a	132

小文字―大文字	10 進数	16 進数	8 進数
a-A=	32	20	40
h-H=	32	20	40
q-Q=	32	20	40
z-Z=	32	20	40

以上より、ある文字の小文字－大文字は一定の値になることがわかった。一定の値とは、10 進数で 32（SP)、16 進数で 20、8 進数で 40 である。（z-Z の 16 進数の計算で 7a-5a というのがあったが、小文字 a をそのまま数字として扱い、引き算ができるかどうか疑問だった。しかし、a-A、h-H、q-Q の計算より 7a-5a=20 と判断。つまり、a-a=0 ということがわかった）

考察 4.2：考察 1.1 により、7a-5a のような文字が入った ASCII コードでも計算できることが判明。ここである疑問が生じた。a-a=0 ならば、a+a=2a という感じで数学的な計算ができるのか.。a-a=0 ができたので a+a=2a もできるはずである。

実験：ここでは 16 進数になおすと『a』がある ASCII コードの計算をする。

（1）j+z

16 進数では 6a+7a になる。10 進数で計算すると答えは 228 である。

まずは普通の数学風でやってみる（ただの足し算)。6a+7a=13a

13a を 10 進数で表すと、1×16 の 2 乗＋3×16 の 1 乗＋10×16 の 0 乗=314 となり、失敗。

ちょっと工夫した方法でやってみる。1 桁目と 2 桁目を分けて計算する。そして 1 桁目の計算結果に 16 の 0 乗を掛け、2 桁目の計算結果に 16 の 1 乗を掛けて足す。

6a+7a=（6+7）×16 の 1 乗＋(a+a)×16 の 0 乗=208+2a

=208+2×10

=228　　←答えと同じ

上の計算は 16 進数を 10 進数に変換する過程の計算を少し変えてみただけだが、今実験中の『2a』が現れており、最終的にでてきた答えが一致している。つまり、a+a=2a という考えは正しいということになる。よって、

a+a は計算でき、2a という答えになる。（桁数を分けた場合であり、4a+a=5a のようなただの足し算はできない）

考察 4.3:『k-A』、『T+q』などの計算を簡単にできる方法はないのか。

アルファベットや、数字を ASCII コードの 10 進数で表した時を考えてみる。

大文字	10 進数	小文字	10 進数	数字	10 進数
A	65	a	97	0	48
B	66	b	98	1	49
C	67	c	99	2	50
…	…	…	…	…	…
Y	89	y	121	8	56
Z	90	z	122	9	57

表示可能な ASCII コードの範囲は 10 進数で 32~126 の範囲。

表示可能な ASCII コードは合計 95 個である。

この表を見てみると、A に 32 を足すと a になる。これは B と b、C と c の関係と同じである。

ASCII コードの計算をするプログラムを作ってみる

ソースコード：

```
#include<stdio.h>

int main(){
  int  i, j, m;

  for(i='A';i<='N';i++){
   j=i+32;
   m=j-i;

   printf("'%c'-'%c'='%c'\t",j,i,m);
   if(i=='N')break;


printf("'%c'-'%c'='%d'\n",j+13,i+13,(j+13)-(i+13));
}
printf("\n");

return(0);
}
```

出力結果：

```
'a'-'A'=' '     'n'-'N'='32'
'b'-'B'=' '     'o'-'O'='32'
'c'-'C'=' '     'p'-'P'='32'
'd'-'D'=' '     'q'-'Q'='32'
'e'-'E'=' '     'r'-'R'='32'
'f'-'F'=' '     's'-'S'='32'
'g'-'G'=' '     't'-'T'='32'
'h'-'H'=' '     'u'-'U'='32'
'i'-'I'=' '     'v'-'V'='32'
'j'-'J'=' '     'w'-'W'='32'
```

'k'-'K'=' ' 'x'-'X'='32'

'l'-'L'=' ' 'y'-'Y'='32'

'm'-'M'=' ' 'z'-'Z'='32'

'n'-'N'=' '

実験

```
#include <stdio.h>
  int main(){
  int p, q ,r;

  for(p='A';p<='Z';p++){
    q='a';
    r=q-p;

    printf("'%c'-'%c'='%d'\t",q,p,r);
    if(p=='Z')break;
    }

  return(0);
  }
```

出力結果：

'a'-'A'='32' 'a'-'B'='31' 'a'-'C'='30' 'a'-'D'='29' 'a'-'E'='28'

'a'-'F'='27' 'a'-'G'='26' 'a'-'H'='25' 'a'-'I'='24' 'a'-'J'='23'

'a'-'K'='22' 'a'-'L'='21' 'a'-'M'='20' 'a'-'N'='19' 'a'-'O'='18'

'a'-'P'='17' 'a'-'Q'='16' 'a'-'R'='15' 'a'-'S'='14' 'a'-'T'='13'

'a'-'U'='12' 'a'-'V'='11' 'a'-'W'='10' 'a'-'X'='9' 'a'-'Y'='8'

'a'-'Z'='7'

これは引き算をやってみた。次は足し算をやってみる。

実験：

```
#include <stdio.h>
  int main(){
  int p, q ,r;
```

```
for(q='A';q<='N';q++)
for(p='A';p<='N';p++)
{
  r=q+p;

  printf("'%c'+'%c'='%d'\t",q,p,r);
  if(q=='N')break;
  if(p=='N')break;
 }

return(0);
}
```

出力結果：

```
'A'+'A'='130'    'A'+'B'='131'    'A'+'C'='132'    'A'+'D'='133'    'A'+'E'='134'
'A'+'F'='135'    'A'+'G'='136'    'A'+'H'='137'    'A'+'I'='138'    'A'+'J'='139'
'A'+'K'='140'   'A'+'L'='141'     'A'+'M'='142'    'A'+'N'='143'    'B'+'A'='131'
'B'+'B'='132'    'B'+'C'='133'    'B'+'D'='134'    'B'+'E'='135'    'B'+'F'='136'
'B'+'G'='137'    'B'+'H'='138'   'B'+'I'='139'     'B'+'J'='140'    'B'+'K'='141'
'B'+'L'='142'    'B'+'M'='143'    'B'+'N'='144'    'C'+'A'='132'    'C'+'B'='133'
'C'+'C'='134'    'C'+'D'='135'    'C'+'E'='136'   'C'+'F'='137'     'C'+'G'='138'
'C'+'H'='139'    'C'+'I'='140'    'C'+'J'='141'    'C'+'K'='142'    'C'+'L'='143'
'C'+'M'='144'    'C'+'N'='145'    'D'+'A'='133'    'D'+'B'='134'   'D'+'C'='135'
'D'+'D'='136'    'D'+'E'='137'    'D'+'F'='138'    'D'+'G'='139'    'D'+'H'='140'
'D'+'I'='141'    'D'+'J'='142'    'D'+'K'='143'    'D'+'L'='144'    'D'+'M'='145'
'D'+'N'='146'    'E'+'A'='134'    'E'+'B'='135'    'E'+'C'='136'    'E'+'D'='137'
'E'+'E'='138'    'E'+'F'='139'    'E'+'G'='140'    'E'+'H'='141'    'E'+'I'='142'
'E'+'J'='143'   'E'+'K'='144'     'E'+'L'='145'    'E'+'M'='146'    'E'+'N'='147'
'F'+'A'='135'    'F'+'B'='136'    'F'+'C'='137'    'F'+'D'='138'    'F'+'E'='139'
'F'+'F'='140'    'F'+'G'='141'   'F'+'H'='142'     'F'+'I'='143'    'F'+'J'='144'
'F'+'K'='145'    'F'+'L'='146'    'F'+'M'='147'    'F'+'N'='148'    'G'+'A'='136'
'G'+'B'='137'    'G'+'C'='138'    'G'+'D'='139'   'G'+'E'='140'     'G'+'F'='141'
'G'+'G'='142'    'G'+'H'='143'    'G'+'I'='144'    'G'+'J'='145'    'G'+'K'='146'
'G'+'L'='147'    'G'+'M'='148'    'G'+'N'='149'    'H'+'A'='137'    'H'+'B'='138'
```

'H'+'C'='139'　'H'+'D'='140'　'H'+'E'='141'　'H'+'F'='142'　'H'+'G'='143'

'H'+'H'='144'　'H'+'I'='145'　'H'+'J'='146'　'H'+'K'='147'　'H'+'L'='148'

'H'+'M'='149'　'H'+'N'='150'　'I'+'A'='138'　'I'+'B'='139'　'I'+'C'='140'

'I'+'D'='141'　'I'+'E'='142'　'I'+'F'='143'　'I'+'G'='144'　'I'+'H'='145'

'I'+'I'='146'　'I'+'J'='147'　'I'+'K'='148'　'I'+'L'='149'　'I'+'M'='150'

'I'+'N'='151'　'J'+'A'='139'　'J'+'B'='140'　'J'+'C'='141'　'J'+'D'='142'

'J'+'E'='143'　'J'+'F'='144'　'J'+'G'='145'　'J'+'H'='146'　'J'+'I'='147'

'J'+'J'='148'　'J'+'K'='149'　'J'+'L'='150'　'J'+'M'='151'　'J'+'N'='152'

'K'+'A'='140'　'K'+'B'='141'　'K'+'C'='142'　'K'+'D'='143'　'K'+'E'='144'

'K'+'F'='145'　'K'+'G'='146'　'K'+'H'='147'　'K'+'I'='148'　'K'+'J'='149'

'K'+'K'='150'　'K'+'L'='151'　'K'+'M'='152'　'K'+'N'='153'　'L'+'A'='141'

'L'+'B'='142'　'L'+'C'='143'　'L'+'D'='144'　'L'+'E'='145'　'L'+'F'='146'

'L'+'G'='147'　'L'+'H'='148'　'L'+'I'='149'　'L'+'J'='150'　'L'+'K'='151'

'L'+'L'='152'　'L'+'M'='153'　'L'+'N'='154'　'M'+'A'='142'　'M'+'B'='143'

'M'+'C'='144'　'M'+'D'='145'　'M'+'E'='146'　'M'+'F'='147'　'M'+'G'='148'

'M'+'H'='149'　'M'+'I'='150'　'M'+'J'='151'　'M'+'K'='152'　'M'+'L'='153'

'M'+'M'='154'　'M'+'N'='155'　'N'+'A'='143'

完成。

以上より、ASCII コードに関してこれだけは覚えておきたいということを、まとめてみた。

1.ASCII コードで表示可能な範囲は 32~126 である

2.ASCII コードの順序は、おおまかにいうと、スペース→数字→英大文字→英小文字

3.10 進数でいうと、32 がスペース（SP）。48 が数字の 0。65 が英大文字の A。

97 が英小文字の a。126 が終了

4.2 番の順序を覚える。そして 10 進数でいう 32,ASCII コードでいうと SP（スペース）と英大文字と英小文字には関係があり、英大文字に SP を足すと同じ英語の小文字になる。また、逆に英小文字から SP を引くと同じ英語の大文字いなる。

例

A+SP=a　　a-SP=A　　a-A=SP

エラー考察：今回は面白い（びっくりしたが）エラーがでたのでぜひ書きたかったので書くことにした。

```
#include <stdio.h>
  int main(){
  int p, q ,r;

  for(p='A';p<='Z';p--){
    q='a';
    r=q-p;

    printf("'%c'-'%c'='%c'\t",q,p,r);
    if(p=='Z')break; }

  return(0);
  }
```

出力結果：

無限ループ。

コンピュータが止まらなくなった。そして強制終了の仕方がわからなく、とりあえずキーボードを触ってなんとか止めようとした。なんとかして止まったのだが、その方法を覚えていない。

原因は for(p='A';p<='Z';p--)の部分だ。この部分を文章になおすと、『初期値は p=A。そして p が Z 以下ならば p の値を 1 ずつ引いていく』。ASCII コードで A は 65。Z は 90。A から 1 ずつ引いてくと、Z の値から遠ざかっていく。これではいつまでたっても Z に近づかない。よってずっと 1 が引かれていく。

解決策としては、p--を p++にする。

ループは初めて体験したが、あせった。

参考文献：

インターネット

http://www.ie.u-ryukyu.ac.jp/ie/students/j06-j.html（情報工学科学生のページ）

http://ja.wikipedia.org/wiki/メインページ（辞典）

感想：

明日テストだ～～～。でも宿題やってない～～。的な感じの 6 月 6 日の水曜日でした。いや～～プログラミングって難しいですね。今回は無限ループも出現してあせったし。自分で考えたプログラムを作ろうとしてもぼ～～っと本を見てただけで結局できなかく、先輩のページをみてソースコードをとってきた。先輩ってすごいなって思わせる水曜日でした。