錬成問題

■　右に示すプログラム部分の誤りを指摘せよ。

(1)

```
char  str[10];

str = "ABC";
printf("%s", str);
```

■　右に示すプログラム部分の実行結果を示せ。

(2)
(3)

```
int    i;
char   str[] = "ABC";
char  *ptr   = "123";

for (i = 0; str[i]; i++)
    putchar(str[i]);
putchar('\n');

for (i = 0; ptr[i]; i++)
    putchar(ptr[i]);
putchar('\n');
```

■　右に示すのは、文字列*st*を空文字列にする関数である（添字演算子を用いずに実現すること）。

```
void null(char *st)
{
    (4)  = '\0';
}
```

■　右に示すのは、文字列*st*が空文字列であれば1を、そうでなければ0を返す関数である（添字演算子を用いずに実現すること）。

```
int is_null(const char *st)
{
    return (    (5)    );
}
```

■　右に示すのは、文字列*st*の内容が"ABC"であれば1を、そうでなければ0を返す関数である（添字演算子を用いずに実現すること）。

```
int isABC(const char *st)
{
    if ( (6) != 'A')  return (0);
    if ( (7) != 'B')  return (0);
    if ( (8) != 'C')  return (0);
    if ( (9) != '\0') return (0);
    return (1);
}
```

■　右に示すのは、文字列*st*を表示する関数である。

```
void putstr(const char *st)
{
    while (*st)
        putchar(  (10)  );
}
```

■ 右に示すプログラム部分の実行結果を示せ。なお、*p*[1]は［ (11) ］が格納されているアドレスで初期化される。

［ (12) ］
［ (13) ］
［ (14) ］
［ (15) ］

```
char  s[3][5] = {"LISP", "C", "Ada"};
char *p[3]    = {"1234", "5", "678"};

putchar(s[0][1]);   putchar('\n');
putchar(s[1][0]);   putchar('\n');
putchar(p[0][2]);   putchar('\n');
putchar(p[2][0]);   putchar('\n');
```

■ 右に示すのは、文字列*st*を後ろの文字から逆順に表示する関数である。

```
void putrstr(const char *st)
{
    char *p = st;

    while ( (16) )
        st++;
    if (st > p)
        while ( (17) > p)
            putchar( (18) );
}
```

■ 右に示すのは、文字列*s1*と*s2*を連続して表示する関数である。

```
void putstrstr(const char *s1, const char *s2)
{
    while (*s1) putchar( (19) );
    while (*s2) putchar( (20) );
}
```

■ 右に示すのは、文字列*s*の長さ（ナル文字は含まない）を返す関数である。

```
int str_len(const char *s)
{
    int  len = 0;

    while ( (21) )
        len++;
    return (len);
}
```

■ 右に示すのは、文字列*s*を、末尾のナル文字も含めて*d*にコピーし、受け取った*d*と同じ値を返す関数である。

```
char *str_cpy(char *d, const char *s)
{
    char  *t = (22) ;

    while (*(23) ++ = *(24) ++)
        ;
    return (t);
}
```

（次ページへ続く）

■　以下に示すのは、文字列*s2*を*s1*にコピーし、受け取った*s1*と同じ値を返す関数である。ただし、文字列*s2*の長さが*n*以上であれば*n*文字までコピーし、*n*より短ければ残りをナル文字で埋め尽くす。

```
char *str_ncpy(char *s1, const char *s2, size_t n)
{
    char *tmp = s1;

    while ( (25) ) {
        n--;
        if (!(*s1++ = *s2++))
            (26) ;
    }

    while (n--)
        *(27)++ = '\0';

    return ( (28) );
}
```

■　右に示すのは、文字列*s1*の後ろに、末尾のナル文字も含めて文字列*s2*を付け加え、受け取った*s1*と同じ値を返す関数である。

```
char *str_cat(char *s1, const char *s2)
{
    char *tmp = s1;

    while ( (29) )
        s1++;

    while (*s1++ = *s2++)
        (30)

    return ( (31) );
}
```

■　以下に示すのは、文字列*s1*の後ろに、文字列*s2*を付け加え、受け取った*s1*と同じ値を返す関数である。ただし、文字列*s2*の長さが*n*より長い場合は切り捨てる。

```
char *str_ncat(char *s1, const char *s2, size_t n)
{
    char *tmp = s1;

    while ( (32) )
        s1++;

    while (n--)
        if (!(*s1++ = *s2++))
            (33) ;

    (34) = '\0';

    return ( (35) );
}
```

■ 右に示すのは、文字列*st*に含まれている数字文字'0'～'9'の個数を、*cnt*[0]～*cnt*[9]に格納する関数である。

```
void cntdigit(const char *st, int cnt[])
{
    int  i;

    for (i = 0; i < 10; i++)
        *(cnt + (36) ) = 0;

    while (*st) {
        if (*st >= '0'  &&  *st <= '9')
            cnt[ (37) ]++;
        (38) ++;
    }
}
```

■ 右に示すのは、数字でない文字を除いて文字列*st*を表示する関数である。

```
void putstrdgt(const char *st)
{
    while ( (39) ) {
        if (*st >= '0'  &&  *st <= '9')
            putchar( (40) );
        (41) ++;
    }
}
```

■ 右に示すのは、文字列*st*の文字の並びを逆順に並べかえる関数である。

```
void rev_string(char *st)
{
    char *pt = st;

    while (* (42) )
        pt++;

    if (pt != (43) ) {
        pt (44) ;
        while (pt > st) {
            char  temp = *pt;
            (45) = *st;
            (46) = temp;
        }
    }
}
```

■ 右に示すのは、文字列*str*から文字*c*を探索し、最も先頭側に位置する文字へのポインタ（見つからない場合はNULL）を返す関数である。

この関数は、どのようなときに意図通りに動作しないかを説明せよ。

(47)

```
char *str_char(const char *str, int c)
{
    for ( ; *str != c; str++) {
        if (*str == '\0')
            return (NULL);
    }

    return ((char *)str);
}
```