プログラミングC　2024 年度期末試験

2024 年 11 月 28 日 9:10-10:40 （90 分）

以下の注意事項を守って問題に解答しなさい。

・解答開始の指示があるまで、このページ以外を見てはならない。

・問題用紙は、表紙を含めて両面刷り 9 枚（全 18 ページ）である。

・途中での提出・退席は 9 時 50 分まで認めない。

・解答用紙は片面刷り 4 枚である。

・解答は所定の場所に、読みやすくきれいに記述すること。

・解答用紙の各ページそれぞれに「学籍番号」「氏名」を記入すること。

・机の上には学生証と筆記用具のみを置くこと。机の中には物を入れてはいけない。

・携帯電話等は電源を切り、鞄の中にしまうこと。着信音がなった場合は不正とみなす。

・問題プログラム中で、例えば「exit(2);」の「(2)」のようなカッコ書き数字は、
問題番号でないので注意すること。

・フォントの都合上、バックスラッシュ「＼」は「¥」として印刷されている。

・実行例ではキーボード入力を***./a.out***のように太字の斜体で表示してある。

・本試験は 120 点満点である。

すべての不正行為に対して、断固たる処置を行なう。

問題 1 （20 点）

コマンドライン引数で指定した入力ファイルから英単語を読み込み、指定した英単語を別の英単語に置き換え、結果を出力ファイルに書き込むプログラムを作成したい。要件は以下の通りである。
・入力ファイルは、英単語のみから構成され、英単語毎に改行されて保存されている。
・英単語の最大の長さは、マクロ L で定義される。

このプログラムは、以下のように動作する。
・コマンドライン引数では、入力ファイル、出力ファイル、置換対象の英単語、置換後の英単語を与える。
・最初に、コマンドライン引数の数の確認を行い、数が合わない場合、エラーメッセージを標準エラー出力に出力し、プログラムを強制終了する。
・次に、入力ファイルと出力ファイルをオープンする。オープンできなければ、エラーメッセージを標準エラー出力に出力し、プログラムを強制終了する。
・次に、入力ファイルから英単語を 1 つずつ読み込み、置換対象の英単語かどうかを strcmp 関数で確認する。置換対象であれば、置換後の英単語を出力ファイルに書き込む。そうでない場合は、読み込んだ英単語をそのまま出力ファイルに書き込む。なお、置換対象の判定の際に大文字と小文字は区別される。
・入力ファイルの最後まで処理が終わったら、プログラムを正常終了する。

以下に示す実行例とプログラム中のコメントを参考に、空欄を適切な語句で埋めよ。
なお、同じ番号の空欄には同じ内容が入るものとする。

【実行例】

```
% cat input.txt
The
University
of
Aizu
in
Japan
% ./a.out
Usage: ./a.out inputfile outputfile oldword newword
% ./a.out input.txt output.txt Aizu
Usage: ./a.out inputfile outputfile oldword newword
% ./a.out input.txt output.txt Aizu Tokyo
% cat output.txt
The
University
of
Tokyo
in
Japan
%
```

次ページに続く

【プログラム】

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define LEN 45

int main(int argc, char *argv[]) {
    FILE *fpin, *fpout;
    char word[LEN]; /* 英単語を入れる文字配列 */

    /* コマンドライン引数の数の確認 */
    if(argc != ____(1)____){
        fprintf(____(2)____, "Usage: ./a.out inputfile outputfile oldword newword¥n");
        exit(1);
    }

    /* 入力ファイルをオープン */
    if((fpin = ____(3)____) == NULL){
        fprintf(____(2)____, "Cannot open %s.¥n", argv[1]);
        exit(2);
    }

    /* 出力ファイルをオープン */
    if((fpout = ____(4)____) == NULL){
        fprintf(____(2)____, "Cannot open %s.¥n", argv[2]);
        ____(5)____; /* 入力ファイルをクローズ */
        exit(3);
    }

    while(1){
        /* 英単語を読み込む */
        if(fscanf(____(6)____) != 1) break;

        /* 置換対象か確認し、出力ファイルへ結果を書き込む */
        if(strcmp(____(7)____) == 0){
            fprintf(____(8)____);
        }else{
            fprintf(____(9)____);
        }
    }

    ____(5)____; /* 入力ファイルをクローズ */
    ____(10)____; /* 出力ファイルをクローズ */

    return 0;
}
```

問題 2（20 点）

問題 2-1

以下に示すプログラムが正しく動作するよう、プログラム中の空欄を適切な語句で埋めた上で、実行例の空欄も適切な語句で埋めよ。

int 型のサイズは 4 バイト、ポインタのサイズは 8 バイトとする。
なお、このプログラムを実際に実行すると、アドレスなどの表示結果は毎回同じになるとは限らず、またその内容も実行例とは異なるが、実行例冒頭の内容が

```
a:      0x14a0
&p:     0x1498
```

だったと仮定した上で回答せよ。アドレスは 16 進数であることにも注意せよ。

【プログラム】

```
#include <stdio.h>
#include <string.h>

int main()
{
  int i;
  int a[4], *p, *q;

  p = q = a;

  printf("a:      %p¥n", a);
  printf("&p:     %p¥n", &p);
  printf("============¥n");

  for( i=0 ; i<4 ; i++, q++ ) {
    *q = i*2;
    printf("i = %d: *q = %d¥n", i, *q);
  }
  printf("============¥n");
  printf("a[2]:    ____(1a)____¥n", a[2]);
  printf("&a[2]:   ____(2a)____¥n", &a[2]);
  printf("p:       ____(3a)____¥n", p);
  printf("p+2:     ____(4a)____¥n", p+2);
  printf("*p+2:    ____(5a)____¥n", *p+2);
  printf("*(p+2):  ____(6a)____¥n", *(p+2));
  printf("a+1:     ____(7a)____¥n", a+1);
  printf("*(a+1):  ____(8a)____¥n", *(a+1));
  printf("q:       ____(9a)____¥n", q);
  printf("&p:      ____(10a)____¥n", &p);

  return 0;
}
```

【実行例】

```
% ./a.out
a:      0x14a0
&p:     0x1498
============
i = 0: *q = 0
i = 1: *q = 2
i = 2: *q = 4
i = 3: *q = 6
============
a[2]:    ____(1b)____
&a[2]:   ____(2b)____
p:       ____(3b)____
p+2:     ____(4b)____
*p+2:    ____(5b)____
*(p+2):  ____(6b)____
a+1:     ____(7b)____
*(a+1):  ____(8b)____
q:       ____(9b)____
&p:      ____(10b)____
%
```

問題 2-2

以下に示すプログラムは、str_array に格納された 4 つの文字列

1. Wakamatsu
2. Hongou
3. Takada
4. Bange

の中から任意の順番で文字列を 4 回選び（同じ物を繰り返し選んでも構わない）、選択順に結合させた文字列として文字配列 combined_str に格納し表示する。

これらを踏まえ、以下の問いに答えなさい。

(1)-(6) 以下に示す実行例とプログラム中のコメントを参考に、空欄を適切な語句で埋めよ。
なお、異なる番号の空欄に同じ内容が入ることもある。

(7) このプログラムにおける str_array は文字列配列と文字列定数配列のいずれか、答えよ。

(8) str_array と同じ方法で確保した文字列の内容は、プログラムの中から書き換え可能か不可能か、答えよ。

【実行例】

```
% ./a.out
Choose word #1: 1
Choose word #2: 2
Choose word #3: 3
Choose word #4: 4
WakamatsuHongouTakadaBange
% ./a.out
Choose word #1: 4
Choose word #2: 2
Choose word #3: 3
Choose word #4: 1
BangeHongouTakadaWakamatsu
% ./a.out
Choose word #1: 1
Choose word #2: 1
Choose word #3: 1
Choose word #4: 1
WakamatsuWakamatsuWakamatsuWakamatsu
%
```

【プログラム】

```
#include <stdio.h>
#include <string.h>

int main() {
    int i, j;     /* ループのカウンタ用変数 */
    int k = 0;    /* combined_str への格納位置を制御する変数 */
    int wordlen;  /* str_array から取り出す文字列の長さを格納する変数 */
    int order[4]; /* 文字列の順番の選択結果を格納する変数 */
```

次ページに続く

```
    char *str_array[]={"Wakamatsu","Hongou","Takada","Bange"};
    char combined_str[100];

    for (i = 0; i < 4; i++) {
        printf("Choose word #%d: ", i+1);
        scanf("%d", &order[i]);
    }

    for (i = 0; i < 4; i++) {
        wordlen = ____(1)____(str_array[____(2)____]);
        for (j = 0; j < wordlen; j++) {
            combined_str[k] = str_array[____(3)____][ ____(4)____];
            ____(5)____;
        }
    }

    combined_str[k] = ____(6)____;
    printf("%s¥n", combined_str);

    return 0;
}
```

問題 3（24 点）

三角形の辺の長さを小さい順に a、b、c としたとき、三平方の定理により

$a^2 + b^2 = c^2$ なら直角三角形、$a^2 + b^2 > c^2$ なら鋭角三角形、$a^2 + b^2 < c^2$ なら鈍角三角形

であることが知られている。

そこで、いくつか（最大 10 個まで）の三角形の頂点の x、y 座標を読み込んで、各三角形の三辺の長さを小さい順に表示するとともに、それが直角三角形か、あるいは鋭角／鈍角三角形であるかを判定して表示するプログラムを作成した。関数の説明は、プログラム中のコメントを参照すること。

以下に示す構造体の仕様、実行例、プログラム中のコメントを参考に、空欄を適切な語句で埋めよ。
なお、同じ番号の空欄には同じ内容が入るものとする。

【使用する構造体について】

・XY 型は平面上の点一つを表す構造体型で、メンバー x とメンバー y（浮動小数点型）をもつ。
・TRIANGLE 型は三角形を表す構造体型で、三角形の各頂点を表すメンバー pt（XY 型の要素 3 個の配列）と、三角形の各辺を表すメンバー side（TSIDE 型の要素 3 個の配列）をもつ。
・TSIDE 型は、三角形の辺一つを表す構造体型で、番号を表すメンバー ind（整数型）と、辺の長さの 2 乗を表すメンバー len2（浮動小数点型）をもつ。なお、番号 ind は辺の長さを計算したときの配列添字である。（途中で辺の長さが小さい順になるようにデータを並び替えるため、もともとの頂点と辺の対応関係が分からなくなるのを避けるために設定されている。今回の問題では表示結果には影響しない。）

【実行例】

```
% ./a.out
頂点の座標を x1 y1 x2 y2 x3 y3 のように入力してください
0 0  4 0  0 3
0 2 -1 0  1 0
4 0  0 4  1 1
Ctrl+D
三角形 1
辺の長さは小さい順に  3.000  4.000  5.000
直角三角形です
三角形 2
辺の長さは小さい順に  2.000  2.236  2.236
鋭角三角形です
三角形 3
辺の長さは小さい順に  3.162  3.162  5.657
鈍角三角形です
%
```

【プログラム】

```
#include <stdio.h>
#include <math.h>
#define  NMAX  10
```

次ページに続く

```
typedef struct { /* 点（平面座標）を表す構造体 */
    double x;
    double y;
}XY;

typedef struct { /* 辺の構造体 */
    int ind;   /* 辺の番号 */
    double len2; /* 辺の長さの２乗 */
}TSIDE;

typedef struct { /* 三角形の構造体 */
    XY pt[3];   /* ３つの頂点 */
    TSIDE side[3];   /* ３つの辺 */
}TRIANGLE;

/* プロトタイプ宣言 */
int inputall(TRIANGLE *);
void calcside(TRIANGLE *);
void sortside(TRIANGLE *);
void print_tritype(TRIANGLE);

int main() {
    TRIANGLE tri[NMAX]; /* 三角形データを入れる配列 */
    int n, i;

    printf( "頂点の座標を x1 y1 x2 y2 x3 y3 のように入力してください¥n" );
    /* 頂点の座標データを入力する関数を呼ぶ */
    ____(1)____ = inputall( ____(2)____ );

    /* 三角形の辺の長さの 2 乗の計算 */
    for ( i=0 ; i<n ; i++ ){
        calcside( ____(3)____ );
    }

    /* 辺の長さをソート */
    for ( i=0 ; i<n ; i++ ){
        sortside( ____(3)____ );
    }

    /* 辺の長さと三角形の種類の判定結果の表示 */
    for ( i=0 ; i<n ; i++ ){
        printf( "三角形 %d¥n", i+1 );
        print_tritype( ____(4)____ );
    }

    return 0;
}
```

次ページに続く

```
/* TRIANGLE 型構造体配列に頂点座標データをすべて入力する関数。
   戻り値は読み込みに成功した TRIANGLE 型構造体配列の要素数（座標ではなく、三角形の数） */
int inputall(TRIANGLE *a){
    int i, n;

    for ( n=0 ; n<NMAX ; n++ ){   /* 三角形ごとのループ */
        for ( i=0 ; i<3 ; i++ ){   /* 3 つの頂点を読み込むループ */
            if (scanf( "%lf %lf", ____(5)____.x, ____(5)____.y ) != 2) return n;
        }
    }
    return n;
}

/* 三角形の辺の長さの 2 乗を計算する関数。
   結果は、引数で指定された構造体の side[]メンバー内に格納する。
   戻り値は無し。 */
void calcside(TRIANGLE *p){
    int i, j;
    double dx, dy;

    for ( i=0 ; i<3 ; i++ ){
        j = (i+1)%3;   /* (i,j)の組は(0,1),(1,2),(2,0)となる */
        dx = ____(6)____.x - ____(7)____.x;   /* 2 点の x 座標の差 */
        dy = ____(6)____.y - ____(7)____.y;   /* 2 点の y 座標の差 */
        p->side[i].len2 = dx*dx+dy*dy;  /* 2 点間の長さの 2 乗の計算 */
        p->side[i].ind = i;   /* 辺の番号（添字 i とする）を記録しておく */
    }
}

/* 辺の長さが小さい順にソートする関数。
   仮引数は TRIANGLE 型構造体のポインタ。
   構造体の side[]メンバー配列の要素を並び替える。
   戻り値は無し。 */
void sortside(TRIANGLE *p){
    int i, j, imin;
    ____(8)____ temp;   /* 辺データの一時退避用 */

    for ( i=0 ; i<2 ; i++ ){
        imin = i;
        for ( j=i+1 ; j<3 ; j++ ){ /* 添字 i+1 以降と比較し最も短い辺を探す */
            if (p->side[imin].len2 >  p->side[j].len2) imin = j;
        }
        /* i 番と imin 番のデータを交換 */
        temp = p->side[i];
        p->side[i] = p->side[imin];
        ____(9)____;
    }
}
```

次ページに続く

```
/* 結果表示の関数 */
/* 辺の長さを表示し、次に三角形の種類（直角／鋭角／鈍角）を判定して表示する。
   引数は TRIANGLE 型構造体。
   戻り値は無し。 */
void print_tritype(TRIANGLE t){
    double a, b, c, a2, b2, c2;

    /* まず辺の長さを表示する処理 */
    a2 = ____(10)____[0].len2;   /* 各辺の長さの２乗 */
    b2 = ____(10)____[1].len2;
    c2 = ____(10)____[2].len2;
    a = sqrt( a2 );   /* 平方根をとる */
    b = sqrt( b2 );
    c = sqrt( c2 );
    printf( "辺の長さは小さい順に  %.3f  %.3f  %.3f¥n", a, b, c );

    /* 辺の長さの 2 乗を用いて、三角形の種類を判定して表示（三平方の定理による） */
    /* 平方根の結果には誤差がありうるので、a, b, c は使わず a2, b2, c2 を使うこと */
    if (____(11)____) printf( "直角三角形です¥n" );
    else if (____(12)____) printf( "鋭角三角形です¥n" );
    else printf( "鈍角三角形です¥n" );
}
```

問題 4（20 点）

xy 平面において、対角線上の 2 点で表された x 軸および y 軸に平行な辺を持つ長方形の面積を求めたい。点のデータは N 個の XY 構造体配列に入力されるものとしてプログラムを 3 つのファイル(q4header.h、q4main.c、q4calc.c)に分けて作成した。q4main.c と q4calc.c は、それぞれ単体でのコンパイルと動作試験を行うことができるようになっている。更に、q4main.c ではデータの与えかたを変えてコンパイルできる。

マクロ関数 PRINT(a,b,r)：a と b は XY 型構造体、r は double 型とし、それらの数値を表示する。
マクロ関数 myabs(x)：三項演算子を利用し、x の絶対値を求める。

これらを踏まえ、以下の問いに答えなさい。

(1)-(10) 以下に示すプログラム中のコメントを参考に、空欄を適切な語句で埋めよ。
　なお、同じ番号の空欄には同じ内容が入るものとするが、異なる番号でも同じ内容が入ることもある。

(11) main 関数の単体テストを行うときのコンパイルコマンドを書きなさい。
　ただし、データはダミーデータを使用するものとする。

(12) rectarea 関数の単体テストを行うコンパイルコマンドを書きなさい。

(13) (12)でコンパイルした実行ファイルを実行したときの出力を書きなさい。

(14) 全体を結合した実行ファイルを作るときのコンパイルコマンドを書きなさい。
　ただし、データの入力はリダイレクション用にすること。
　なお、1 つのコマンドでも実現可能だが、複数行に分けてもよい。

【プログラム】

```
==== q4header.h ====
/* マクロ定義 */
#define N 5
#define PRINT(a,b,r) printf("(%.1f, %.1f)-(%.1f,%.1f): area=%.1f¥n", _______(1)_______)
#define myabs(x)  (____(2a)____? ____(2b)____: ____(2c)____)

/* XY 構造体 */
typedef struct {
    double x,y;
} XY;

/* プロトタイプ宣言 */
double rectarea(XY, XY);

==== q4main.c ====
#include <stdio.h>
#include ____(3)____

int main() {
    int i;
    double res;
```

次ページに続く

```
#____(4)____ INPUT==1  /* ダミーデータで data[N]を初期化する場合 */
    XY data[N]={{1.0,1.0},{2.0,2.0},{3.0,1.0},{5.0,2.0},{0.0,0.0}};
#____(5)____ INPUT==2  /* リダイレクションにより data[]に値を入力する場合 */
    XY data[N];
    for (i=0; i<N; i++) {
        scanf("%lf%lf",&data[i].x, &data[i].y);
    }
#_____(6)____  /* キーボードから data[]に値を入力する場合（メッセージ付き) */
    XY data[N];
    printf("点データ(x,y)を%d セット入力して下さい¥n",N);
    for (i=0; i<N; i++) {
        scanf("%lf%lf",&data[i].x, &data[i].y);
    }
#____(7)____

    for (i=0; i<N-1; i++) {
        res=rectarea(data[i], data[i+1]);
        PRINT(__________(8)__________);
    }
    return 0;
}

#____(9)____ DEBUG
double rectarea(XY a, XY b) {
    printf("DEBUG 用 rectarea 呼出¥n");
    PRINT(a,b,99.9);
    return 99.9;
}
#____(10)____

==== q4calc.c ====
#include <stdio.h>
#include ____(3)____

#____(9)____ DEBUG
int main() {
    XY p1={0.0,0.0},p2={2.0,2.0};
    double res;

    res=rectarea(p1, p2);
    PRINT(p1, p2, res);
    return 0;
}
#____(10)____

double rectarea(XY a, XY b) {
    return myabs((a.x-b.x)*(a.y-b.y));
}
```

問題 5（16 点）

自然数の中で不思議な性質を持つものとして、カプレカ数が存在する。
「ある数字の桁を並べ替えて最大にした数と最小にした数との差を取る操作」をカプレカ操作と呼び、例外（1111 などの 4 桁がゾロ目の数字）を除いてどんな 4 桁の数字（3 桁以下の場合は 0 で埋める。例えば 1 なら 0001 として、4 桁にして扱う。）に対しても、この操作を繰り返すと「6174」に辿り着く。この数字をカプレカ数と呼び、この 4 桁のカプレカ数にカプレカ操作を行うと元の数字「6174」となる。
以下に示す実行例では、「2024」の 4 桁数字に対してカプレカ操作を行い「3996」という数字が得られる。続けて、その数字にカプレカ操作を行うことで「6264」の数字が得られる。同様に、この操作を繰り返すとカプレカ数「6174」に到達する。
以下に示すプログラムは、1~9999 の任意の 4 桁以下の数字を標準入力で受け取り、カプレカ操作を再帰的に行いカプレカ数に到達するまでの過程を表示するプログラムである。関数の説明は、プログラム中のコメントを参照すること。

以下に示す実行例とプログラム中のコメントを参考に、空欄を適切な語句で埋めよ。
なお、同じ番号の空欄には同じ内容が入るものとする。

【実行例】

```
% ./a.out
Please input number (1~9999): 2024
4220 - 0224 = 3996
9963 - 3699 = 6264
6642 - 2466 = 4176
7641 - 1467 = 6174
7641 - 1467 = 6174
func "kaprekar" was called 5 times.
%
```

【プログラム】

```
#include<stdio.h>
#include<stdlib.h>

/*プロトタイプ宣言*/
int kaprekar(int);
int get_max_number(int);
int get_min_number(int);
void number2digit(int, int *);
void sort(int *, int);

int main() {
    int number;
    int repeat;
    printf("Please input number (1~9999): ");
    scanf("%d",&number); /*標準入力から数字を受け取る*/
```

次ページに続く

```
    if(number<1 || number > 9999){  /*0 以下の整数、または 5 桁以上の数字は対象外*/
        printf("Please Input number (1~9999)¥n");
        exit(1);
    }

    repeat = kaprekar(number); /*カプレカ操作をする関数の呼び出し*/
    printf("func ¥"kaprekar¥" was called %d times.¥n",repeat);

    return 0;
}

/* kaprekar 関数
引数に、カプレカ操作を行う数字を受け取る。
ゾロ目の場合の例外を除いて、再帰的に呼び出す。
引数とカプレカ操作後の値が同じなら再帰終了。
戻り値は、kaprekar 関数が呼ばれた回数。*/
int kaprekar(int number){
    int max_number; /*桁の数字を並べ替えて最大にした数字を保持する用*/
    int min_number; /*桁の数字を並べ替えて最小にした数字を保持する用*/
    int result;/*最大-最小の計算結果を保持する用 */
    static int repeat = 0; /*kaprekar 関数が呼ばれた回数をカウントする変数*/

    repeat++;

    /*桁の数字を並べ替えて最大の数字を求める関数の呼び出し*/
    max_number = ____(1)____ ;
    /*桁の数字を並べ替えて最小の数字を求める関数の呼び出し*/
    min_number = ____(2)____ ;
    /*最大から最小を引く*/
    result = ____(3)____ ;

    /*過程の表示*/
    printf("%04d - %04d = %04d¥n",max_number,min_number,result);

    /*ゾロ目だった場合の例外処理*/
    if(result == 0){
        printf("Exception was detected.¥n");
        exit(2);
    }
    /*（再帰）計算結果がこの関数の引数の値と異なれば、再帰的に関数を呼び出す。*/
    else if(____(4)____){
        return ____(5)____ ;
    }
    /*（再帰終了）計算結果がこの関数の引数の値と同じ場合、kaprekar 関数が呼ばれた回数を返す。*/
    else return ____(6)____;
}
```

次ページに続く

```
/* get_max_number 関数
整数型数字を受け取り、桁の数字を並べ替えて最大にした数字を返す関数 */
int get_max_number(int number){
    int digits[4];
    /*4 桁数字の各桁の数字を取得する number2digit 関数を呼び出す。*/
    ____(7)____ ;
    /*各桁の数字が格納されている配列をソートする sort 関数を呼び出す。*/
    ____(8)____ ;

    number = digits[3]*1000+digits[2]*100+digits[1]*10+digits[0];
    return number;
}

/* get_min_number 関数
整数型数字を受け取り、桁の数字を並べ替えて最小にした数字を返す関数 */
int get_min_number(int number){
    int digits[4];
    /*4 桁数字の各桁の数字を取得する number2digit 関数を呼び出す。*/
    ____(7)____ ;
    /*各桁の数字が格納されている配列をソートする sort 関数を呼び出す。*/
    ____(8)____ ;

    number = digits[0]*1000+digits[1]*100+digits[2]*10+digits[3];
    return number;
}

/* number2digit 関数
引数に、整数型変数と整数型配列の 2 つを受け取る。
第一引数の整数を桁ごとの数字に分解し、第 2 引数で受け取った配列に保持する。
この関数の第一引数は、4 桁以下の自然数に限定している。
例 1) 第一引数が 2024 の場合、第二引数の配列に[2,0,2,4]が格納されるように処理される。
例 2) 第一引数が 321 の場合、第二引数の配列に[0,3,2,1]が格納されるように処理される。
戻り値なし。*/
void number2digit(int number, int* digits){
    /*処理内容は省略*/
}

/* sort 関数
引数は、整数型配列と配列の要素数を表す整数型変数
整数型配列の内容を昇順に並べ替える
戻り値はなし。*/
void sort(int *array, int size){
    /*処理内容は省略*/
}
```

問題 6（20 点）

以下のような構造体で構成されるノードからなる連結リストがある。
先頭ノード head は外部変数として作成され、連結リストの最後のノードの next は NULL になっている。
既に複数のノードで構成された連結リストが存在しているものとして、連結リスト中の各ノードの学籍番号とキーボードから入力した学籍番号の一部が一致するノードすべてについて、それらのデータを表示する listprint_psel 関数を、クイズ 12 で作成した部分一致関数 partial_match を利用して作成しなさい。ただし、連結リストの検索は「head の次のノードから NULL の前までたどる」という方法で実現すること。
partial_match 関数は、第 1 引数 str で与えられた文字列の中に第 2 引数 m で与えられた文字列が含まれているかを調べる（部分一致）関数で、str 内の一致した箇所のアドレスを返し、一致するものが無かった場合は NULL を返す。

プロトタイプ宣言は以下のように与えられる（説明の都合上、仮引数も記載した）。
また、partital_match 関数は完成していて、内部で必要な関数は与えられていると考えてよい。
なお、実行例の結果表示は表示位置を合わせているが、解答では合わせなくてもよく、表示すべき項目がもれなく表示されていればよい。
アロー演算子が使えるところは必ず使うこと。

```
void listprint_psel(char *sid);
char *partial_match(char *str, char *m);

/* Record 構造体の宣言 */
typedef struct {
    char id[10];   /* 学籍番号 */
    char name[12]; /* 名前 */
    int age;       /* 年齢 */
} Record;

/* node 構造体の宣言 */
typedef struct node *NodePointer;
struct node {
    Record data;
    NodePointer next;
};
```

【実行例】

```
% ./a.out
Head -
  1301001 Taro         19
  1301022 Hanako       22
    : 省略
  1301164 Yoshiko      18
  1301200 Masaru       19
14 nodes exist in the list.
```

次ページに続く

```
Input match data(ID) -> 100
  1301001 Taro        19
  1301004 Jiro        18
  1301007 Ichiro      20

Input match data(ID) -> 12
  1301123 Hikaru      19
  1301200 Masaru      19

Input match data(ID) -> Ctrl+D
%
```

【ソースプログラム】

```
int main() {
    Record r;
    int i;
    FILE *fp;
    char sid[20];

    /* リストの初期化&読み込み(ノードの挿入） 省略 */

    listprint();

    /* input match data(ID) */
    /* Ctrl+D でループを抜ける */
    while (1) {
        printf("Input match data(ID) -> ");

        if (scanf("%s", sid) != 1) {
            printf("¥n");
            break;
        }
        listprint_psel(sid);
    }
    printf("¥n");
    return 0;
}

void listprint_psel(char *sid) {
    NodePointer n; /* 変数などは追加しないこと */

    /******************/
    /* ここを作成する */
    /******************/

    printf("¥n");
}
```