プログラミングＣ 平成 28 年度期末試験

平成 29 年 2 月 10 日

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
- この期末試験は100点満点である
- 解答時間は 80 分間とする
- 退席は開始後 40 分まで認めない
- 問題用紙はこの表紙を含めて両面刷り9枚（全18ページ）である
- 解答用紙は片面刷り2枚である。解答は所定の場所に記述すること
- 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
- 筆記具のみを机の上に置くことができる
- 机の中には一切の物を入れてはいけない
- 携帯電話等は電源を切り、鞄の中に入れておくこと。もし鳴動した場合は不正とみなす
- 実行例中、「%」はコマンドプロンプト、太字斜体の文字はキーボードからの入力を示す
- フォントの都合上、'¥n' などの「¥」はバックスラッシュ「＼」を表わす。
- 解答用紙はスペースの関係で解答欄の順序が分かりにくくなっているので注意すること（特に問題１、２、３）

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（(1-1)各1点x10+(1-2)2=12点）

以下は、テキストファイルを読み込み、行ごとに文字数をカウントし、更に行番号を付加して行毎に表示するプログラムである。以下の２つの問いに答えよ。

(1-1) コメントを参考に空欄に適切な語句を入れてプログラムを完成させよ。同じ番号には同じ語句が入る。なお、exit()の中の数字はプログラムの一部であり、解答欄の番号ではない。また、ファイル名はコマンドラインで与え、与えられない場合は、ファイル `in.txt` を使用するものとする。エラーの場合は標準エラー出力にメッセージを出力する。

(1-2) ファイル`q1b.txt`が以下の内容である時（紙面の都合で改行されているように見えるが、この文章には改行がないとする）`./a.out q1b.txt` を実行した時、画面上に表示される出力をすべて書け。

[ファイル`q1b.txt`の内容]

```
The mission of the University of Aizu is "to Advance Knowledge for Humanity," or
in other words, to make discoveries and inventions which will contribute to the
peace and prosperity of people.
```

[プログラム]

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

#define NF 128
#define NB 100

int main(int argc, char *argv[])
{
  FILE *fp;                    /* ファイルポインタの宣言 */
  int  i, n, c;                /* n:行数 c:読み出した文字  */
  char infile[NF] = "in.txt";  /* 入力ファイルの初期値 */
  char buf[NB];                /* 一行分のファイル読み出しデータ */

  if(argc == 2) { /* ファイル名がコマンドラインで指定されている */
    if(___(1)___(___(2)___) + 1 > NF){  /* ファイル名が長すぎる */
      fprintf(___(3)___, "File name: %s is too long¥n", ___(2)___);
      exit(6);
    }
    strcpy(infile, ___(2)___); /* 指定された入力ファイル名を設定 */
  }

  if((fp = ___(4)___) == NULL){ /* ファイルオープンとエラー処理 */
    fprintf(___(3)___, "%s open error¥n", infile);
    exit(7);
  }
```

```
  for(n = 1;  ; n++) {                    /* nをカウントアップする無限ループ */
    for(i = 0; i < ___(5)___; i++) {      /* 1行読み込みループ */
      c = ___(6)___;                      /* 一文字読み込み */
      if(c == ___(7)___ || c == '¥n') { /* ファイル終端または改行判定 */
        buf[i] = ___(8)___ ;    /* 文字列の終端文字を代入 */
        ___(9)___;              /* ループ脱出 */
      }
      buf[i] = (char)c; /* 通常の場合は読み出した文字をbufに格納 */
    }
    if(i == ___(5)___){ /* 一行の行数がbufの大きさを超えた */
      fprintf(___(3)___, "Buffer overflow!¥n");
      exit(8);
    }
    printf("%3d %3d %s¥n", n, ___(1)___(buf), buf); /* 行番号、長さ、行内容表示 */
    if(c == ___(7)___) ___(9)___; /* ファイルの終端の場合はループ脱出 */
  }

  ___(10)___; /* ファイルクローズ */
  return 0;
}
```

[実行結果]

```
% cat q1a.txt
The mission of the University of Aizu is "to Advance Knowledge
for Humanity," or in other words, to make discoveries
and inventions which will contribute to the peace and
prosperity of people.%
% ./a.out q1a.txt
  1  63 The mission of the University of Aizu is "to Advance Knowledge
  2  54 for Humanity," or in other words, to make discoveries
  3  54 and inventions which will contribute to the peace and
  4  21 prosperity of people.
%
```

問題２（各2点x9=18点）

以下のプログラムがある時、次の２問に答えよ。

(2-1) プログラムの実行結果を書け。番号自体の記入は不要で、結果は解答用紙のそれぞれの番号の所に記入せよ。なお、初めに自分が使用するコンピュータの種別（Solaris, Mac）のどちらかに丸をつけ、その機種での実行結果を書くこと。

(2-2) (9)の部分には動的メモリ割り当てで３×Ｎ個の文字型二次元配列を確保する文が入る。適切な文を書け。キャストは不要である。

[プログラム]

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define N 15

int main()
{
  char str1[][N] = {"Mercury", "Venus", "Jupiter"};
  char *str2[]   = {"Saturn", "Uranus", "Neptune"};
  char *ptr1;
  char (*ptr2)[N];
  int  i;

  printf("(1) %d¥n", (int)sizeof(str1));
  printf("(2) %d¥n", (int)sizeof(str2));

  ptr1 = &str2[1][2];
  printf("(3) %c¥n", *ptr1 + 1);
  printf("(4) %c¥n", *(ptr1 + 1));
  printf("(5) %s¥n", ptr1 + 1);

  ptr2 = ________(9)_________;
  for(i = 0; i < 3; i++) strcpy(ptr2[i],str2[i]);

  printf("(6) %s¥n", &ptr2[2][4]);
  printf("(7) %c¥n", ptr2[2][1] + 1);
  printf("(8) %s¥n", ptr2[0] + 1);

  free(ptr2);
  return 0;
}
```

問題３（各1点x20=20点）

以下の二つの問いに答えよ。

(3-1) 標準入力から３次元空間の点の座標をいくつか（最大 10 個、それぞれ x, y, z 座標の順に並べたもの）を読み込み、すべての二点の組み合わせについて二点間の距離を計算した後、距離が最大となる組のその距離と座標を表示するプログラムを作成したい。なお簡単のため、入力データは全て異なった点であり、少なくとも２点入力され、読み込み時のエラーはないものとする。

点の座標は(x, y, z)３つのメンバを持つ構造体を用いて表し、これを Point 型と名付ける。また二つの点の座標とその間の距離を表すため、Point 型ポインタ変数２つと、二点間の距離を格納する double 型変数１つをメンバに持つ構造体 Dinfo 型を作ってこれを使用する。

実行例とプログラム中のコメントを参考にして、意図した動作をするようにプログラム１の空欄を埋めなさい。なお、同じ番号の空欄には同じ答えが入る。

(3-2) プログラム１の dcalc 関数は、int 型変数 i, j, k を使わずに、Point 型ポインタ変数 q, r を使ってプログラム２（dcalc 関数部分のみ示す）のように書き換えることができる。空欄を埋めて、dcalc 関数がプログラム１の場合と同様の動作をするようにしなさい。ただし他に変数宣言を追加してはならない。

[実行例]

```
% ./a.out
Input coordinates:
1.0 3.0 1.0
5.0 0.0 1.0
2.0 4.0 1.0
-1.0 2.0 -1.0
^D
Maximum distance: 6.633250
between (5.000 0.000 1.000) and (-1.000 2.000 -1.000)
%
```

[プログラム1]

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

_____(1)_____ { /* ３次元空間の点を表す座標の構造体 */
  double x;
  double y;
  double z;
} Point;

_____(1)_____ { /* 二つの点の情報とその間の距離を格納する構造体 */
  double dst;     /* 二点間の距離 */
  Point *p1;      /* 一つの点へのポインタ */
  Point *p2;      /* 一つの点へのポインタ */
} Dinfo;

void dcalc(Point *, Dinfo *, int);
double distance(Point, Point);
```

```
int main()
{
  Point p[10];   /* 点の数は最大10個を仮定 */
  Dinfo di[45];  /* 10C2=45 なので二点の組合せは最大45組 */
  Dinfo dimax;   /* 距離最大となる二点の情報を入れる構造体 */
  int i, n;

  printf("Input coordinates:¥n");
  for ( i=0; i<10; i++ ){
    if ( scanf( "%lf%lf%lf", ___(2)___, ___(3)___, ___(4)___ ) != 3 ) break;
  }
  n = i;

  /* 距離計算等を行う関数を呼ぶ */
  dcalc( _____(5)_____ );

  /* 距離の最大値を探す。探した結果はdimax構造体に記録 */
  dimax.dst = 0.0;
  for ( i=0; i<n*(n-1)/2; i++ ){
    if (_____(6)_____ > dimax.dst) {
        _____(7)_____;
    }
  }

  /* 結果表示：距離の最大値と二点の座標を表示 */
  printf("Maximum distance: %f¥n", dimax.dst);
  printf("between  (%.3f  %.3f  %.3f)  and  (%.3f  %.3f  %.3f)¥n",  ____(8)____x,
____(8)____y, ____(8)____z, ____(9)____x, ____(9)____y, ____(9)____z);

  return 0;
}

/* Point型の配列データから、二つの点の組み合わせ全てについて二点間の
   距離を計算し、Dinfo型の配列に距離と二つの点の情報を格納する。
   なお個々の距離の計算は別の関数を呼んで行う。第3引数は点の数。 */
void dcalc(Point *p, Dinfo *d, int n)
{
  int i, j, k=0;

  for ( i=0; i<n-1; i++ ){   /* 片方の点の添え字 */
    for ( ___(10)___; j<n; j++ ){   /* もう一方の点の添え字 */
      d[k].dst = distance( _____(11)_____ ); /* 距離計算関数を呼ぶ */
      d[k].p1 = ____(12)____; /* 計算に使われた点へのポインタを格納 */
      d[k].p2 = ____(13)____;
      k++;
    }
  }
}
```

```
/* 二つの点を引数にとり、その間の距離を計算して返す関数 */
double distance(Point p1, Point p2)
{
  Point v;

  v.x = p2.x-p1.x;   /* いったん座標値の差を新しい変数に格納 */
  v.y = p2.y-p1.y;
  v.z = p2.z-p1.z;
  return sqrt( _______________(14)_______________ );
}
```

[プログラム２]

```
void dcalc(Point *p, Dinfo *d, int n)
{
  Point *q, *r;

  for ( q=p; q<p+n-1; q++ ){   /* q が&p[0]から&p[n-2]まで変化 */
    for ( ___(15)___; ___(16)___; r++ ){ /* r が&p[n-1]まで変化 */
      d->dst = distance( _____(17)_____ ); /* 距離計算関数を呼ぶ */
      ___(18)___ = q;   /* 計算に使われた点へのポインタを格納 */
      ___(19)___ = r;
      ___(20)___;       /* 配列の要素を一つ先に */
    }
  }
}
```

問題４（3点ｘ5点=15点）

以下は標準入力から入力されたテキスト中の単語数と行数を数えるために作成したプログラムだが、単語数を数える関数はファイル Q4_wordc.c に、それ以外の処理はファイル Q4_main.c に別々に作成されている。各ファイルは単体での動作確認試験と、両方を合わせた全体での動作確認試験ができるようになっている。また、UNIX 系のテキスト（改行は'¥n'で表される）を扱う場合と、Windows 系で作られたテキスト（改行は'¥r'と'¥n'の 2 文字一組で表される）を扱う場合とで、2 種類の異なる実行ファイルを作れるようになっている。(なお、プリプロセッサ関連の部分を除き、プログラムは既に正しく作成されていると仮定してよい。)
この時以下の４つの問題に答えよ。

(4-1) Q4_wordc.c 単体で動作確認のためにコンパイルする時の UNIX コマンドを答えよ。

(4-2) 二つのプログラムを合わせた全体で動作確認試験する際には、実行結果を例えば「Lines: 20 Words: 40」のように出力するが、Q4_main.c 単体でコンパイルし動作確認する際は、単語数はカウントされないので、実行結果を「Lines: 20」のように出力したい。意図した動作となるよう Q4_main.c の空欄(A)(B)を埋めよ。ただし、動作確認に使用するためのマクロ名は(4-1)と同じものを使用するものとする。

(4-3) 二つのファイルを合わせて全体でコンパイルする際、Q4_main.c の【ア】の領域が実行ファイルに含まれるようにコンパイルする場合の UNIX コマンドを１行で答えよ。

(4-4) /* 【イ】 */を付けた行と、その次行にわたる if else 文を、if 等を使わず三項演算子を使って１行で書き換えよ。

[プログラム Q4_main.c]

```
#include <stdio.h>

int wordc(char);

int main()
{
  char c;
  int tmp;   /* getchar()の戻り値を一時的に格納 */
  int n=0, w=0, r_flg=0;

  while ( (tmp = getchar()) != EOF  ){ /* 文字読み込み */
    c = (char) tmp;
#ifdef WINDOWS
    if (r_flg==1 && c == '¥n') { /* r_flg は一つ前の文字が'¥r'かどうかのフラグ */
      n++;              /* 改行文字のカウント（Windows 系） */
      r_flg=0;
    }
    if (c == '¥r') r_flg=1;   /* 【イ】 */
    else r_flg=0;
#else
    if (c == '¥n') n++;   /* 改行文字のカウント（UNIX 系） */
#endif
    w += wordc( c );
  }
```

【ア】

```
  printf("Lines: %d", n);   /* 結果出力 */
_____(A)_____
  printf("   Words: %d", w);
_____(B)_____
  printf("¥n");
  return 0;
}

/* 単体テスト用のダミーの wordc 関数 */
#ifdef DEBUG
int wordc(char c)
{
  return 0;
}
#endif
```

[プログラム Q4_wordc.c]

```
#include <stdio.h>

int wordc(char);

#ifdef DEBUG
int main()   /* 単体テスト用に main も作成した */
{
  int tmp;
  int w=0;

  while ( (tmp = getchar()) != EOF  ) w += wordc( (char)tmp );
  printf("Words: %d¥n", w);
  return 0;
}
#endif
```

```
/* スペース、タブ、改行で区切られた単語数を数える関数。
   新しい単語が始まる時だけ 1 を返し、それ以外の時は 0 を返す */
int wordc(char c)
{
  /* w_flg は区切り文字を読み込んだら 0、それ以外を読んだら 1 に切り替わるフラグ */
  static int w_flg=0;

  /* スペース、タブ、改行（'¥n'に加え Windows で使われる'¥r'も）のチェック */
  if (c == ' ' || c == '¥t' || c == '¥n' || c == '¥r') {
    w_flg=0;
    return 0;
  }
  else if (w_flg == 0) { /* 単語の始まり（区切り文字の後に普通の文字）か？ */
    w_flg=1;
    return 1;
  }
  else return 0;
}
```

問題５（各1点x15=15点）

以下の２つの問題に答えよ。

(5-1) フィボナッチ数列は、次に示す漸化式で定義される。

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}\ (n \geq 2)$$

以下は、マクロＮで指定した項数までフィボナッチ数列を求めて表示するプログラムである。プログラム中で関数 `fibonacci` を用いて再帰的に数列の各項の値を求めている。また、関数 `fibonacci` は、自身の呼び出し回数を記録し、表示している。空欄を埋めてプログラムを完成させよ。

[実行例]

```
% ./a.out
Function fibonacci() is called 1 times.
F0 = 0
Function fibonacci() is called 2 times.
F1 = 1
(中略)
F4 = 3
(中略)
Function fibonacci() is called 34 times.
F5 = 5
%
```

[プログラム]

```
#include <stdio.h>
#define N 5

int fibonacci(int);

int main()
{
  int i;

  for(i = 0; i <= N; i++){
    printf("F%d = %d¥n",i,fibonacci(i));
  }

  return 0;
}
```

```
int fibonacci(int n) {
  ____(1)____ int k = ____(2)____;
  printf("Function fibonacci() is called %d times.¥n", k);
  ____(3)____
  if (____(4)____) {
    return n;
  } else {
    return ____(5)____+____(6)____;
  }
}
```

(5-2) 以下は、ベクトルを正規化する（ベクトルの長さを 1 にする）関数 vector_normalize を用いたプログラムの例である。

関数 vector_normalize はアドレス渡しで正規化前・正規化後のベクトルを受け渡しする。第一引数が正規化前、第二引数が正規化後のベクトルが格納される配列へのポインタである。また、二次元、三次元、さらに高次元のベクトルでも対応できるように、ベクトルの次元を第三引数で受け取る。戻り値は正規化前のベクトルの長さである。空欄を埋めてプログラムを完成させよ。なお、同じ番号の空欄には同じ内容が入る。

[参考]

n 次元ベクトル $\vec{a} = (a_1, a_2, a_3, \ldots, a_n)$ を正規化したベクトル $\hat{\boldsymbol{a}}$ は、$\hat{a} = \dfrac{\vec{a}}{|\vec{a}|}$ で得られる。ただし、

$$|\vec{a}| = \sqrt{\sum_{i=1}^{n} a_i^2}$$

[実行例]

```
% ./a.out
Norm of vector2D: 2.828
  Normalized vector2D[0]: 0.707
  Normalized vector2D[1]: 0.707

Norm of vector3D: 2.449
  Normalized vector3D[0]: 0.408
  Normalized vector3D[1]: 0.816
  Normalized vector3D[2]: 0.408
%
```

[プログラム]

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

double vector_normalize(____(1)____, ____(1)____, int);

int main()
{
  int i;
  double vector2D[2] = {2.0, 2.0}; /* 2 次元ベクトルの例 */
  double vector2D_n[2];
  double vector3D[3] = {1.0, 2.0, 1.0}; /* 3 次元ベクトルの例 */
  double vector3D_n[3];

  printf("Norm of vector2D: %.3f¥n", vector_normalize(vector2D, vector2D_n,
____(2)____));
  for (i = 0; i < 2; i++) {
    printf("  Normalized vector2D[%d]: %.3f¥n",i , vector2D_n[i]);
  }
  printf("¥n");
  printf("Norm of vector3D: %5.3f¥n", vector_normalize(vector3D, vector3D_n,
____(3)____));
  for (i = 0; i < 3; i++) {
    printf("  Normalized vector3D[%d]: %.3f¥n",i , vector3D_n[i]);
  }

  return 0;
}

double vector_normalize(____(1)____in, ____(1)____out, int n){

  double norm = 0;
  int i;

  /* ベクトルの長さの計算 */
  for (____(4)____) {
    ____(5)____ += ____(6)____*____(6)____;
  }

  ____(5)____ = sqrt(____(5)____); /* sqrt()は平方根を計算する関数 */

  /* 正規化 */
  for (____(4)____) {
    ____(7)____ = ____(8)____;
  }

  ____(9)____;
}
```

問題６（各2点x10=20点）

以下に示すのは連結リストによって、学生の学籍番号、氏名、成績を管理するためのプログラムである。ただし、いくつかの関数については内容を省略してある。
連結リストの各ノードは、講義および演習で取り扱った連結リストと同様の仕様で、node 型の自己参照型構造体である。node 型構造体のメンバは学生情報を格納する Record 型の構造体 data と、次ノードを示すポインタ next であり、リストの終端ノードでは、next にはヌルポインタ（NULL）が入っているものとする。このプログラムはプログラム１に、その時の実行例は実行例１に示されている。この時、以下の４つの問いに答えよ。

(6-1) 連結リストの最初のノードは、ヘッドノードと呼ばれる特別なノードから始まっている。ヘッドノードは他のノードと同じ型の構造体であるが、次に続くノードへの連結の情報のみを保持している。また、プログラム中の全ての関数から参照出来るようになっている。この機能を実現するためにヘッドノードのための NodePointer 型の変数 head の宣言をプログラムのどこで、どのように行えばよいか。宣言の場所として適切な場所を、プログラム中に記された宣言位置候補(a)-(c)から選んで記号で答えよ。

(6-2) finditem 関数は、連結リスト内を検索し、引数として与えられた学籍番号 id に一致する情報を持つノードを検索する。ただし、当該ノードそのものではなく、その一つ前のノード位置を返す。関数内の空欄 2-1～2-4 を埋めよ。

(6-3) delete 関数は、引数として与えられた学籍番号 id に一致する情報を持つノードを削除する機能を持つ。具体的には、まず finditem 関数を利用して削除対象ノード delnode の一つ前のノード位置を得て、次いでノード間の連結を組み替え、最後に削除対象ノード delnode のメモリを解放する。関数内の空欄 3-1～3-2 を埋めよ。

(6-4) insert 関数は、Record 型の引数として与えられた学生の学籍番号、氏名、成績の情報を格納した新しいノードをヘッドノードの直後に挿入する。この関数を書き換えて、連結リストのノード並びが学籍番号順になるようなノードの挿入を行う insert_sort 関数を作成したい。プログラム２で示された insert_sort 関数の空欄 4-1～4-3 を埋めよ。なお、同じ番号の空欄には同じ内容が入る。insert 関数ではなく、insert_sort 関数を使用した場合の実行例を実行例２に示す。

[実行例１]

```
% ./a.out
Insert new data: (ID name score) -> 1221010 Hatanaka 67
Head -
   1221010 Hatanaka    67

Insert new data: (ID name score) -> 1231164 Egawa 61
Head -
   1231164 Egawa       61
   1221010 Hatanaka    67

Insert new data: (ID name score) -> 1231022 Watanabe 90
Head -
   1231022 Watanabe    90
   1231164 Egawa       61
   1221010 Hatanaka    67
```

```
Insert new data: (ID name score) -> 1231200 Higashio 61
Head -
   1231200 Higashio    61
   1231022 Watanabe    90
   1231164 Egawa      61
   1221010 Hatanaka    67

Insert new data: (ID name score) -> ^D
Delete ID -> 1231022
Head -
   1231200 Higashio    61
   1231164 Egawa      61
   1221010 Hatanaka    67

Delete ID -> ^D
%
```

[実行例２]

```
% ./a.out
Insert new data: (ID name score) -> 1221010 Hatanaka 67
Head -
   1221010 Hatanaka    67

Insert new data: (ID name score) -> 1231164 Egawa 61
Head -
   1221010 Hatanaka    67
   1231164 Egawa      61

Insert new data: (ID name score) -> 1231022 Watanabe 90
Head -
   1221010 Hatanaka    67
   1231022 Watanabe    90
   1231164 Egawa      61

Insert new data: (ID name score) -> 1231200 Higashio 61
Head -
   1221010 Hatanaka    67
   1231022 Watanabe    90
   1231164 Egawa      61
   1231200 Higashio    61

Insert new data: (ID name score) -> ^D
Delete ID -> ^D
%
```

[プログラム1]

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

/* 学生の学籍番号、氏名、成績を格納する構造体 */
typedef struct {
  int ID;
  char name[11];
  int score;
} Record;

typedef struct node *NodePointer;

/* 連結リストのノード用の構造体 */
struct node {
  Record data;
  NodePointer next; /* 終端ノードでの next の値は NULL */
/* ヘッドノード宣言位置(a) */
};

NodePointer insert(Record);
NodePointer delete(int);
NodePointer finditem(int);
void listprint(void);
NodePointer make_1node(Record, NodePointer);

/* ヘッドノード宣言位置(b) */

int main()
{
  int i;
  Record r;

/* ヘッドノード宣言位置(c) */

  head = make_1node( r, NULL );

  while (1) {
    printf("Insert new data: (ID name score) -> ");
    if (scanf("%d %s %d", &r.ID, r.name, &r.score) != 3) {
      printf("¥n");
      break;
    }
    if (insert(r) == NULL) {
      printf("Data %d is already on the list!¥n", r.ID);
    }
    listprint();
  }
```

```
  while (1) {
    printf("Delete ID -> ");
    if (scanf("%d", &i) != 1) {
      printf("¥n");
      break;
    }
    if (delete(i) == NULL) {
      printf("Data %d is not found.¥n", i);
    }
    listprint();
  }

  return 0;
}

NodePointer finditem(int id)
{
  NodePointer n;

  /* ノード検索のためのループ */
  for(____(2-1)____; ____(2-2)____; ____(2-3)____) {

    /* ノード検索 */
    if(____(2-4)____ == id) return n;
  }
  return NULL;
}


NodePointer delete(int id)
{
  NodePointer n, delnode;

  /* 削除対象ノードdelnodeの一つ前のノード位置を調べる */
  if ((n = finditem(id)) != NULL){

    /* ノード間の連結を組み替える */
    ____(3-1)____;
    ____(3-2)____;

    /* 削除対象ノードdelnodeのメモリを解放 */
    free(delnode);
  }

  return n;
}
```

```
NodePointer insert(Record r)
{
  NodePointer newnode;

  if (finditem(r.ID) != NULL) return NULL;

  newnode = make_1node(r,head->next);
  head->next = newnode;
  return newnode;
}

void listprint(void)
{
/* listprint関数は、連結リストの内容を表示する */
/* 具体的な関数の内容は省略 */
}

NodePointer make_1node(Record r, NodePointer p)
{
/* make_1node関数は、引数として与えられた情報を内容に持つノードを */
/* 新たに作成し、その位置を戻り値として返す */
/* 具体的な関数の内容は省略 */
}
```

[プログラム２]

```
/* insert_sort関数 */
/* 連結リストのノード並びが学籍番号順になるようなノードの挿入を行う */
NodePointer insert_sort(Record r)
{
  NodePointer newnode, n; /* 新たに必要となる変数nを追加 */

  if (finditem(r.ID) != NULL) return NULL;

  /* ノード挿入位置を検索するためのループ */
  for(____(4-1)____) {

    /* ノード挿入位置の検索 */
    if (____(4-2)____  > r.ID ) break;
  }

  /* ノードの作成とリストへの挿入 */
  newnode = make_1node(r,____(4-3)____);
  ____(4-3)____ = newnode;
  return newnode;
}
```