プログラミングＣ　2023 年度期末試験

2023 年 12 月 5 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 10 枚（全 20 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 3 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、問題番号では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イタリックで表示してある

・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 （24 点）

絶対値 30 以内の整数型データが格納されたファイルがある。そのファイルを読み込み、実行例のようなグラフを表示するプログラムを作成したい。実行例やプログラム中のコメントを参考に空欄を埋めて完成させなさい。"|"はゼロ軸を表しており、それを中心に左側はマイナス、右側はプラスを表している。また、ゼロ軸の左側にはマクロ N で与えた 30 点が描画可能である。

[ファイル内容表示と実行例]

```
% cat data2.txt
20
10
-10
-20
0
5
-7
-30
5
30
25
% ./a.out
Usage: ./a.out filename
% ./a.out data2.txt
[ 20]:                                |********************
[ 10]:                                |**********
[-10]:                     **********|
[-20]:          ********************|
[  0]:                                |
[  5]:                                |*****
[ -7]:                        *******|
[-30]:*******************************|
[  5]:                                |*****
[ 30]:                                |******************************
[ 25]:                                |************************
```

[プログラム]

```
#include <stdio.h>
#include <stdlib.h>
#define N 30

void cgraph(int);

int main(int argc, ____(1)____argv[])
{
```

```
    int data, status;
    FILE ____(2)____;

    if (____(3)____) {
        printf("Usage: %s filename¥n", ____(4)____);
        exit(1);
    }
    fp=fopen(____(5)____);
    if (____(6)____) {
        printf("file open error¥n");
        exit(2);
    }

    while(1) {
        status=__________(7)__________; /* データを読み込む */
        if (____(8)____) break;
        printf("[%3d]:",data);
        cgraph(data);
    }
    fclose(fp);

    return 0;
}

void cgraph(int x)
{
    int i;
    x+=N;
    if (x<N) {
        for (i=0; ____(9)____; i++) printf(" ");
        for (i=x; ____(10)____; i++) printf("*");
        printf("|");
    } else {
        for (i=0; ____(11)____; i++) printf(" ");
        printf("|");
        for (i=N; ____(12)____; i++) printf("*");
    }
    printf("¥n");
}
```

問題２（20 点）

会津大演習室の CentOS（64 ビット）環境で、以下のプログラムを実行した際の出力結果を答えよ。アドレスは 16 進数であることに注意すること。なお、先頭部分の出力はすぐ下に示す通りとする。与えられた情報では出力結果を特定できない場合は、解答欄に×と記入すること。(1)などの問題番号部分は改めて書く必要は無い。

[出力の先頭部分]

```
% ./a.out
int_arr:  0x70a0
str_arr1: 0x7060
str_arr2: 0x7040
```

[プログラム]

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

int main() {
  int  int_arr[] = {1, 10, 100, 1000};
  int  *int_ptr;
  char str_arr1[][20] = {"AizuWakamatsu","Bandai",    "Inawashiro"};
  char *str_arr2[] =    {"AizuBange",    "AizuMisato","Kitakata"  };
  char *char_ptr;
  int  i;

  printf("int_arr:  %p¥n", &int_arr[0]);
  printf("str_arr1: %p¥n", &str_arr1[0]);
  printf("str_arr2: %p¥n", &str_arr2[0]);           /* ここまでは上の実行例の通り */

  printf("( 1) %d¥n", (int)sizeof(int_arr));
  printf("( 2) %d¥n", (int)sizeof(str_arr1));
  printf("( 3) %d¥n", (int)sizeof(str_arr2));

  printf("( 4) %p¥n", int_arr);
  printf("( 5) %p¥n", &str_arr1[1]);
  printf("( 6) %p¥n", &str_arr2[1]);

  int_ptr = &int_arr[2];
  printf("( 7) %d¥n", *(int_ptr + 1));
  printf("( 8) %d¥n", *(int_ptr) + 2);

  str_arr1[2][5] = '¥0';
  printf("( 9) %d¥n", (int)strlen(str_arr1[0]));
  printf("(10) %c¥n", str_arr1[0][5]);
  printf("(11) %s¥n", &str_arr1[1][3]);
```

```
  printf("(12) %d¥n", (int)strlen(str_arr1[2]));

  char_ptr = str_arr2[1];
  printf("(13) %s¥n", *(str_arr2 + 1));
  printf("(14) %s¥n", char_ptr);
  printf("(15) %p¥n", char_ptr);
  printf("(16) %s¥n", char_ptr + 1);
  printf("(17) %c¥n", *(char_ptr + 3));
  printf("(18) %c¥n", *(char_ptr) + 2);

  char_ptr = (char *)malloc(30 * sizeof(char));
  strcpy(char_ptr, str_arr1[0]);
  printf("(19) %d¥n", (int)sizeof(char_ptr));
  printf("(20) %d¥n", (int)strlen(char_ptr));

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

問題３（20 点）

大学の授業を管理するためのプログラムを作成したい。要件は以下の通りである。

・学生の情報を保存する構造体（student）と、講義の情報を保存する構造体（lecture）を作る。
・構造体 student は学生の氏名と成績を保持する。
・構造体 lecture は講義の名称とその講義を受ける学生一人一人の情報を student 型の構造体配列で保持する。
・lecture 型構造体が格納できる学生の情報の最大数（50）はマクロ MAX_STUDENT で定義する。

構造体の内容をまとめると以下の通りである。
<学生: student>
・名前（文字型配列: 最大 20 文字）: name
・成績（int 型）: score
<講義: lecture>
・講義の名称（文字型配列: 最大 20 文字）: lec_name
・学生情報（student 構造体の配列: 最大 50 人）: student_data
・読み込めた学生の数（int 型）: num_student

このプログラムは、以下のように動作する。
・main 関数内で、講義の情報を保存する構造体変数を定義する。
・input_info 関数は、宣言した lecture 構造体変数のアドレスを受け取り、標準入力から受け取った講義名と学生情報を構造体に格納する。学生情報が正しく入力された回数をカウントし、Ctrl-D が入力されるか、または入力内容に問題があった場合には入力を終了する。
・print_info 関数で、講義名とその lecture 構造体の student 型配列に格納されたデータを表示する。
・sorting_by_score 関数で、lecture 構造体変数の中の student 型配列と学生の数を受け取り、その中の **score を基準にして降順（大きい方から小さい方へ）に並べ替える。**
・main 関数の最後で、student 型配列に格納されているデータの score の最大、最小を表示する。

以下に示すプログラムとその実行例を見て、空欄を埋めよ。
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。
また、実行例では 5 人分の例を示したが、入力人数に依存せずに正しく動作するように内容を埋めよ。

【実行例（斜体はキーボード入力）】

```
% ./a.out
Course Name: ProgC
New Student Info (Name, Score): Aizu_Ichiro 65
New Student Info (Name, Score): Aizu_Jiro 80
New Student Info (Name, Score): Aizu_Saburo 90
New Student Info (Name, Score): Aizu_Shiro 20
New Student Info (Name, Score): Aizu_Goro 55
```

```
New Student Info (Name, Score): Ctrl+D
====================================
Course Name : ProgC
====================================
Student name    Score
------------------------------------
Aizu_Ichiro     65
Aizu_Jiro       80
Aizu_Saburo     90
Aizu_Shiro      20
Aizu_Goro       55
====================================
Sorting By Score
====================================
Course Name : ProgC
====================================
Student name    Score
------------------------------------
Aizu_Saburo     90
Aizu_Jiro       80
Aizu_Ichiro     65
Aizu_Goro       55
Aizu_Shiro      20
====================================
Max:90  Min:20
```

【プログラム】

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

#define MAX_STUDENT 50

____(1)____ student {
    char name[20];
    int score;
};

____(1)____ lecture {
    char lec_name[20];
    ____(2)____ student_data[MAX_STUDENT]; /* student 型配列 student_data を定義する */
    int num_student;
};

void input_info(struct lecture *);
void print_info(struct lecture *);
```

```
void sorting_by_score(struct student[], int);

int main(){
    int i;
    ____(3)____ Prog1; /* lecture 型構造体変数 Prog1 を定義する*/

    input_info(____(4)____);
    print_info(____(4)____);
    printf("Sorting By Score¥n");
    sorting_by_score(Prog1.student_data, Prog1.num_student);
    print_info(____(4)____);
    printf("Max:%d¥t",Prog1.student_data[0].score)        /*最大値の表示*/
    printf("Min:%d¥n",Prog1.student_data[____(5)____].score);  /*最小値の表示*/
    return 0;
}

void input_info(struct lecture *p){
    int i;
    printf("Course Name: ");
    scanf("%s",____(6)____); /*講義名の読み込み*/
    for(i=0;i<MAX_STUDENT;i++){
        printf("New Student Info (Name, Score): "); /*学生の名前と点数の読み込み*/
        if(scanf("%s %d",p->student_data[i].name, ____(7)____)!=2) {
           printf("¥n");
           break;
        }
    }
    p->num_student = i; /*読み込めた人数を記録*/
}

void print_info(struct lecture *lec){
    int i;
    printf("=====================================¥n");
    printf("Course Name : %s¥n",lec->lec_name);
    printf("=====================================¥n");
    printf("Student name¥tScore¥n");
    printf("-------------------------------------¥n");
    for(i=0;i<____(8)____;i++){      /*人数分ループを回す*/
        printf("%s¥t%d¥n",lec->student_data[i].name, lec->student_data[i].score);
    }
    printf("=====================================¥n");
}

void sorting_by_score(struct student student_data[], int num_student){
    int i,j;
    struct student tmp;

    for(i=0; i<num_student-1; i++){
```

```
        for(j=i+1; j<num_student; j++){
           /* i 番目の学生の score と j 番目の学生の score を比較して、*/
           /*後者の方が高ければ処理を行う*/
            if(____(9)____ < ____(10)____){
                tmp = student_data[i];
                student_data[i]=student_data[j];
                student_data[j]=tmp;
            }
        }
    }
}
```

問題４（16 点）

問題４－１

２つの複素数を入力し、そのかけ算を行うプログラムを作成する。
プログラムは３つのファイル Q4.h, Q4_main.c, Q4_func.c からなり、それぞれヘッダーファイル、main 関数のファイル、その他の関数のファイルである。全体をコンパイルすると、本来の動作をする実行ファイルが生成される。
Q4_main.c，Q4_func.c はそれぞれ単体テストができるように作成されており、Q4_main.c の単体テストではマクロ DEBUG を定義しておくことで、呼び出した関数の引数と戻り値が確認できる。また、Q4_func.c の単体テストではマクロ CTEST が定義されているときは関数 comp_calc のテスト、マクロ PTEST が定義されているときは関数 print_complex のテストができる。
以上を踏まえ、以下の問題に答えなさい。

（ア）プログラム中の空欄に適切な語句を入れなさい。ただし、同じ番号の空欄には同じ内容が入る。
（イ）それぞれのオブジェクトファイルを作成し、それらを結合した実行ファイルを作成する命令を３行で書きなさい。
（ウ）print_complex 関数の単体テストをしたいときのコンパイルコマンドを書きなさい。
（エ）上記 print_complex 関数の単体テストプログラムを実行したときの表示を書きなさい。

[実行例（全結合コンパイル後）]

```
% ./a.out
複素数 a の実部と虚部を入力して下さい: 1 2
1+i2
複素数 b の実部と虚部を入力して下さい: -3 -4
-3-i4
a*b= 5-i10
%
```

[実行例（main の単体テスト）]

```
% ./a.out
複素数 a の実部と虚部を入力して下さい: 5 6
print_complex:
引数の実部、虚部=5, 6
複素数 b の実部と虚部を入力して下さい: 7 8
print_complex:
引数の実部、虚部=7, 8
comp_calc:
第 1 引数の実部、虚部=5, 6
第 2 引数の実部、虚部=7, 8
a*b= print_complex:
引数の実部、虚部=5, 6
%
```

[実行例（comp_calc の単体テスト）]

```
% ./a.out
(1+i2)*(3+i4)の計算結果は実部-5, 虚部 10 の複素数になる
%
```

[プログラム]

[Q4.h]

```
typedef struct {
  int real;
  int img;
} Complex;

Complex comp_calc(Complex, Complex);
void print_complex(Complex);
```

[Q4_main.c]

```
#include <stdio.h>
#include "Q4.h"

int main()
{
  Complex a,b,c;

  printf("複素数 a の実部と虚部を入力して下さい: ");
  scanf("%d%d", &a.real, &a.img);
  print_complex(a);
  printf("複素数 b の実部と虚部を入力して下さい: ");
  scanf("%d%d", &b.real, &b.img);
  print_complex(b);

  c=comp_calc(a,b);
  printf("a*b= ");
  print_complex(c);
}

#____(1)____ DEBUG
Complex comp_calc(Complex x, Complex y)
{
  printf("comp_calc:¥n");
  printf("第 1 引数の実部、虚部=%d, %d¥n",x.real, x.img);
  printf("第 2 引数の実部、虚部=%d, %d¥n",y.real, y.img);
  return x;
}
```

```
void print_complex(Complex x)
{
  printf("print_complex:¥n");
  printf("引数の実部、虚部=%d, %d¥n",x.real, x.img);
}
#____(2)____
```

[**Q4_func.c**]

```
#include <stdio.h>
#include "Q4.h"

#____(1)____ CTEST
int main()
{
  Complex a={1,2},b={3,4},c;
  c=comp_calc(a,b);
  printf("(1+i2)*(3+i4)の計算結果は");
  printf("実部%d, 虚部%d の複素数になる¥n",c.real, c.img);

  return 0;
}
#____(3)____ PTEST
int main()
{
  Complex a[]={{1,2},{-3,0},{0,4},{0,-1},{5,-2},{0,0}};
  int i;
  for (i=0; i<6; i++) print_complex(a[i]);
  return 0;
}
#____(2)____

Complex comp_calc(Complex x, Complex y)
{
  Complex z;
  z.real=x.real*y.real-x.img*y.img;
  z.img=x.real*y.img+y.real*x.img;

  return z;
}

void print_complex(Complex x)
{
  if (x.img==0) printf("%d ",x.real);
  else if (x.real==0&&x.img>0) printf("i%d", x.img);
  else if (x.real==0&&x.img<0) printf("-i%d", -x.img);
```

```
  else if (x.img<0) printf("%d-i%d",x.real, -x.img);
  else printf("%d+i%d",x.real, x.img);
  printf("¥n");
}
```

問題４－２

変数が `int a, b, *p;` と宣言されており、`a, b` には正の整数が代入されているとする。三項演算子（条件演算子）を使って、次の動作をする代入文を書きなさい。なお、`if` 文は使わないこと。

「`a` と `b` が等しいとき `a` のアドレスを、等しくないとき `b` のアドレスを、`p` に代入する。」

問題 5 （20 点）

原点 O と点 A(x_1, y_1)、点 B(x_2, y_2)の 3 点からなる三角形 △ OAB の面積は

$$\triangle OAB = \frac{1}{2}|x_1y_2 - x_2y_1|$$

で求めることができる。
以下のプログラムは、この公式を利用して三角形の面積を求めている。三角形の表現方法と、三角形の面積を求める関数の仕様の組み合わせにより、以下に示す 3 通りの方法を用いている。

1．三角形を原点と二つの Point 型構造体変数に格納された頂点で表現し、その面積を CalcTriangleArea_1 関数で計算
2．三角形を原点と Point 型構造体配列に格納された頂点二つで表現し、その面積を CalcTriangleArea_2 関数で計算
3．三角形を Triangle 型構造体で表現し、その面積を CalcTriangleArea_3 関数で計算

実行例を参考にして、空欄を埋めてプログラムを完成させよ。同じ番号の空欄には同じ内容が入る。
なお、引数として与えた double 型の実数の絶対値を返す関数 fabs()を用いてよい。fabs()のプロトタイプ宣言は

```
double fabs(double);
```

であるが、以下の例のようにそのまま使用できる。

[fabs()の使用例]

```
x = -1.2;
printf("%f¥n", fabs(x));
  は
1.2
  と表示される（x は double 型の変数）。
```

[実行例]

```
% ./a.out
Area (1): 0.500000
Area (2): 0.500000
Area (3): 0.500000
```

[プログラム]

```
#include <stdio.h>
#include <math.h> /* fabs()の使用のためにインクルード */

typedef struct {
  double  x; /* x 座標 */
  double  y; /* y 座標 */
} Point;
```

```
typedef struct {
  Point  vertex[2];   /* 2 頂点の座標を格納するメンバ */
  double area;        /* 面積を格納するメンバ */
} Triangle;


double CalcTriangleArea_1(_____(1)_____, _____(1)_____);
double CalcTriangleArea_2(_____(2)_____);
void   CalcTriangleArea_3(_____(3)_____);

int main() {
   Point A = {1.0, 2.0};
   Point B = {2.0, 3.0};
   Point AB[2] = {{1.0, 2.0}, {2.0, 3.0}};
   Triangle OAB;

   /* 原点と二つの Point 型構造体変数に格納された頂点 A, B からなる三角形の面積を計算 */
   printf("Area (1): %f¥n", CalcTriangleArea_1(A, B));

   /* 原点と Point 型構造体配列 AB に格納された頂点二つからなる三角形の面積を計算 */
   printf("Area (2): %f¥n", CalcTriangleArea_2(AB));

   /* Triangle 型構造体変数 OAB に格納された三角形の面積を計算 */
   OAB.vertex[0] = A;
   OAB.vertex[1] = B;
   CalcTriangleArea_3(_____(4)_____);
   printf("Area (3): %f¥n", OAB.area);

   return 0;
}

double CalcTriangleArea_1(_____(1)_____ A, _____(1)_____ B){
    _____(5)_____;
}

double CalcTriangleArea_2(_____(2)_____ AB){
    _____(6)_____;
}

void CalcTriangleArea_3(_____(3)_____ OAB){
    _____(7)_____area = _____(8)_____;
}
```

問題 6 （20 点）

以下のプログラムは、学生の成績データを標準入力で受け取り、連結リストで**点数が降順（大→小の順）になるように**連結・挿入し、毎回リストの内容を表示するプログラムである。入力された学籍番号が既にリストに存在する場合は、挿入せずにメッセージを表示する。また、データ入力時に適切なデータが入力されなかった場合にプログラムは終了するものとする。
連結リストの構造は第 9 回のハンドアウトと同様であるが、int 型メンバ key の代わりに以下のメンバを持つ Student 型構造体を使用する。また、node 構造体も以下の通りである。

[Student 型構造体]
- 学籍番号(char 型配列): id[10]
- 点数(int 型): score

[node 構造体]
- 格納データ（Student 型）: data
- 連結先のアドレスを持つポインタ（node のポインタ）: next

[関数の仕様]
・insert 関数
　引数で与えられたデータの学籍番号がすでにリストに存在する場合は、挿入せずに NULL を返す。存在しない場合は、点数が降順になるように挿入を行う。点数が同じ場合は、先に入力されたデータが前に来るように挿入する。
・find_item 関数
　ハンドアウトの仕様とは異なり、同じ id を見つけた場合そのノードを指すポインタを戻り値とし、同じ id が見つからなかった場合は NULL を戻り値とする。
・make_1node 関数
　引数の Student 型構造体と NodePointer をメンバとする node を作成する。
・print_list 関数
　リストをたどりながら data を表示する。

プログラムの下線部を、上に書いた仕様やプログラム中のコメントにしたがい、適切なコードで埋めよ。なお、同じ番号の下線部には同じ答えが入る。

【実行例（*斜体*は入力）】

```
% ./a.out
New node data: s1234567 80
==========================
Head –
s1234567 – 80
==========================
New node data: s1111111 90
```

```
==========================
Head –
s1111111 - 90
s1234567 – 80
==========================
New node data: s2222222 90
==========================
Head –
s1111111 – 90
s2222222 – 90
s1234567 – 80
==========================
New node data: s1234567 50
s1234567 is already existed.
New node data: Ctrl+D
%
```

【プログラム】

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

typedef struct{
    char id[10];
    int score;
}Student;

typedef struct node * NodePointer;

struct node{
    Student data;
    NodePointer next;
};

NodePointer make_1node(Student, NodePointer);  /* ノード作成関数 */
NodePointer find_item(char *); /* 学籍番号検索関数 */
void insert(Student);  /* 新しいデータの挿入関数 */
void print_list(void);  /* リストの内容表示関数 */

NodePointer head;

int main(){
    Student data;

    head = make_1node(data,NULL); /*head の初期化*/
```

```
    while(1){
        printf("New node data: ");
        if(scanf("%s %d",___(1)___, ___(2)___)!=2) break;
        insert(data);
        print_list();
    }
    return 0;
}

void insert(Student newdata){
    NodePointer np, newnode;

    np = find_item(___(3)___);
    if(np != ___(4)___){
        printf("%s is already existed¥n", newdata.id);
    }
    else{
        /*挿入位置の探索*/
        for(np=___(5)___; np->next!=NULL; ___(6)___){
            if(np->next->data.score < newdata.score) break;
        }
        newnode = make_1node(newdata, ___(7)___);
        ___(7)___ = newnode;
    }
}

/* 同じ id が見つかったら、そのノードを指すポインタを返す */
/* 見つからなければ、NULL を返す */
NodePointer find_item(char *target){
    NodePointer np;
    for(np=head->next; np!=NULL; np=np->next){
        if(___(8)___ == 0) return np;
    }
    return NULL;
}

NodePointer make_1node(Student data, NodePointer p){
    NodePointer newnode;

    if((newnode = (___(9)___)malloc(___(10)___))==NULL){
        printf("Error in memory allocation¥n");
        exit(8);
    }
    newnode->data = data;
    newnode->next = p;
    return newnode;
}
```

```
void print_list(void){
    NodePointer np;
    printf("=====================¥n");
    printf("Head -¥n");
    for(np=head->next; np!=NULL; np=np->next){
        printf("%s - %d¥n", np->data.id, np->data.score);
    }
    printf("=====================¥n");
}
```