プログラミングC 2022年度期末試験

2022年12月5日[月] 11:00-12:30（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。

・問題用紙はこの表紙を含めて両面刷り9枚（全17ページ）である。

・途中での提出・退席は11時40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。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。

・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 （22 点）

以下のプログラムは、コマンドライン引数で指定した入力ファイルからテキストを読み込み、英単語のみを出力ファイルに書き出すとともに、英単語の長さの統計をとって、文字数別に何個あったかを画面に表示する。
ここで英単語とはアルファベットが 1 文字以上連続しているものをいい、アルファベット以外（記号や数字、スペース等々）があるときは単語の切れ目と見なす。同じ単語が複数出てくる場合もそれぞれ別々にカウントする。入力・出力ファイル両方が指定されていないときは、メッセージを表示して終了する。
実行例とプログラム中のコメントを参考に、最適な語句で空欄を埋め、プログラムを完成させよ。同じ空欄番号には同じ内容が入る。
なお、string.h 関連の関数は使用しないものとする。

[実行例]

```
% cat Aizu1993.txt   （入力ファイルを見る）
The university of Aizu was established at Aizu-wakamatsu in 1993.
% ./a.out
Usage: ./a.out inputfile outputfile
% ./a.out Aizu1993.txt output.txt
 2 character word: 3
 3 character word: 2
 4 character word: 2
 9 character word: 1
10 character word: 1
11 character word: 1
% cat output.txt   （出力ファイルを見る）
The university of Aizu was established at Aizu wakamatsu in
%
```

[プログラム]

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

#define LEN 32   /* 一単語の最大文字数の設定 */

int main(int argc, _____(1)_____ argv[])
{
  _____(2)_____ *fpin, *fpout;
  int frq[LEN];   /* 文字数別の単語カウンタ用配列。要素[0]は使用しない。 */
  int i, n;
  char c;

  if (_____(3)_____){   /* 入力ファイルと出力ファイルが指定されていない場合は終了 */
    fprintf(stderr, "Usage: ./a.out inputfile outputfile\n" );
    exit (2);
  }
```

次ページに続く

```
  if ((fpin = _____(4)_____) == NULL){
    fprintf(stderr, "Cannot open input file %s!\n", argv[1] );
    exit (3);
  }

  if ((fpout = _____(5)_____) == NULL){
    fprintf( stderr, "Cannot open output file %s!\n", argv[2] );
    _____(6)_____;     /* オープン済みのファイルはクローズ */
    exit (4);
  }

  for ( i=1; i<LEN; i++ ) frq[i] = 0;   /* 単語数をゼロに初期化 */


  n=0;
  while (fscanf(_____(7)_____) == 1 ){ /* ファイルから c に 1 文字読み込み */
    if (_____(8)_____){   /* c が英字を表すなら */
      n++;                  /* 連続する英字をカウント */
      fputc(_____(9)_____);   /* ファイルに 1 文字出力 */
    }
    else {
      if (n > 0){           /* 英字の後に英字以外が来たら単語の切れ目 */
        _____(10)_____;         /* 長さ n 文字の単語としてカウント */
        n=0;                /* 単語長さを 0 にリセット */
        fprintf(fpout, " ");
      }
    }
  }
  fprintf(fpout, "\n");
  _____(6)_____;
  _____(11)_____;     /* ファイルをクローズ */

  for ( i=1; i<LEN; i++ ){
    if ( frq[i] > 0) {     /* 単語数が 1 個以上のときのみ表示 */
      printf( "%2d character word: %d\n", i, frq[i] );
    }
  }

  return 0;
}
```

問題 2（24 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（(1)などの問題番号部分は改めて書く必要は無い）。

[プログラム]

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

int main(){

    int array1[10]={1,2,3,5,8,13,21,34,55,89};
    int *ptr1;

    char array2[3][20]={"UnivOfFukushima","UnivOfAizu","UnivOfTokyo"};
    char *array3[] = {"UnivOfFukushima","UnivOfAizu","UnivOfTokyo"};
    char *ptr2,*ptr3;

    int *ptr4;

    ptr1 = array1;
    printf("(1) %d\n",*(ptr1+5));
    printf("(2) %d\n",*ptr1+5);
    printf("(3) %d\n",(int)sizeof(array1));

    printf("(4) %d\n",(int)sizeof(array2[0]));
    printf("(5) %d\n",(int)strlen(array2[0]));
    printf("(6) %d\n",(int)sizeof(array3[0]));
    printf("(7) %d\n",(int)strlen(&array3[1][6]));


    ptr2 = &array2[1][6];
    printf("(8) %c\n",*ptr2);
    printf("(9) %s\n",ptr2);
    strcpy(&array2[2][4],ptr2);
    printf("(10) %s\n",array2[2]);

    ptr3 = array3[1];
    if(strcmp(ptr3,array2[1])==0)printf("(11) matched\n");
    else printf("(11) unmatch\n");
```

次ページに続く

```
    ptr4 = (int *)malloc(sizeof(int)* 10); /*int 型 10 個分のメモリ領域を確保*/
    printf("(12) %d\n",(int)sizeof(ptr4)); /*メモリ領域の確保後の ptr4 のサイズ*/
    free(ptr4);                            /*確保したメモリ領域の解放*/

    return 0;
}
```

問題３（30 点）

小テストの結果ファイル（欠席情報を含む）をリダイレクションで読み込み、各個人の平均点と出席判定を表示するプログラムを作成したい。
ファイルには学籍番号（文字列）と複数回の小テストの点数（整数）が入っている。ただし、欠席した回の点数は負になっている。これらのデータを扱うために Student 型構造体を利用し、main 関数内で、構造体配列として宣言する。関数 input は、構造体配列の 1 要素のアドレスを引数として、1 要素分のデータを読み込む。データがない場合、EOF を戻す。読み込んだ点数が負の場合、該当メンバーの値は 0 としておき、欠席回数 status をインクリメントする。関数 output は構造体配列とデータの個数を引数として受け取り、学籍番号、小テストの点数、平均点、出席判定（欠席が全回の 3 分の 1 未満なら OK、以上なら NG）を表示する。学生の最大数はマクロ N、小テストの回数はマクロ QNO で与え、平均点は小数点第 2 位で四捨五入するものとする。
実行例とプログラム中のコメントを参考に、最適な語句で空欄を埋め、プログラムを完成させよ。同じ空欄番号には同じ内容が入る。

[実行例]

```
% cat studata.txt
u2204040   62  79  90  76  95  97  68  97 100 100
u2204100   87 -99  95  60 -99 -99 -99  79  93 100
u2210200   95 -99  85  76   8 -99  36  75 -99  95
u2204108   74  88  58  80  54 100  44 -99  96 100
u2210116  100 100 100 100 100 100 100 100 100 100
u2204200  100  81  95  88  96  86  92  97  96  90
u2204188  100  36  34 -99  86  64 -99  97  66 100
u2204007   88  95  57  92  81  92  96  97  88  90
u2210130  100 100 100 100 100  96 100 100 100 100
u2210177  100 100 100 100 100  76  80 100  80  95
% ./a.out < studata.txt
u2204040   62  79  90  76  95  97  68  97 100 100   86.4 OK
u2204100   87   0  95  60   0   0   0  79  93 100   51.4 NG
u2210200   95   0  85  76   8   0  36  75   0  95   47.0 OK
u2204108   74  88  58  80  54 100  44   0  96 100   69.4 OK
u2210116  100 100 100 100 100 100 100 100 100 100  100.0 OK
u2204200  100  81  95  88  96  86  92  97  96  90   92.1 OK
u2204188  100  36  34   0  86  64   0  97  66 100   58.3 OK
u2204007   88  95  57  92  81  92  96  97  88  90   87.6 OK
u2210130  100 100 100 100 100  96 100 100 100 100   99.6 OK
u2210177  100 100 100 100 100  76  80 100  80  95   93.1 OK
%
```

次ページに続く

[プログラム]

```
#include <stdio.h>
#define N 300
#define QNO 10

typedef struct {
  char sid[10];
  int score[QNO];
  double ave;
  int status;
} Student;

int input(_____(1)_____);
void output(_____(2)_____, int);

int main()
{
  Student data[N];
  int i,j,n;

  for (i=0; i<N; i++) {
    if (input(_____(3)_____) == EOF) break;
  }
  n=i;

  /* 平均点の計算 */
  for (i=0; i<n; i++) {
    _____(4)_____ = 0.0; /* 平均点の初期化 */
    for (j=0; j<QNO; j++) {
      _____(4)_____ += _____(5)_____;
    }
    _____(4)_____ /= QNO; /* 平均点算出 */
    _____(4)_____ = _____(6)_____; /* 四捨五入 */
  }

  output(_____(7)_____, n);

  return 0;
}
```

次ページに続く

```
int input(_____(1)_____stu)
{
  int i;
  if (scanf("%s", _____(8)_____)==EOF) return EOF;
  _____(9)_____ = 0; /* 欠席回数の初期化 */
  for (i=0; i<QNO; i++) {
    scanf("%d", _____(10)_____);
    if (_____(11)_____ < 0) { /* 欠席データの処理 */
      _____(11)_____ = 0;
      _____(9)_____++; /* 欠席回数の更新 */
    }
  }
  return 0;
}

void output(_____(2)_____stu, int n)
{
  int i,j;
  for (i=0; i<n; i++) {
    printf("%s  ", _____(12)_____);
    for (j=0; j<QNO; j++) {
      printf("%3d ", _____(13)_____);
    }
    printf(" %5.1f ", _____(14)_____);
    if (_____(15)_____ <= QNO/3) printf("OK\n");
    else printf("NG\n");
  }
}
```

問題４（12 点）

以下のプログラムは、キーボードから平面座標３つの入力を求め、それを三角形の頂点座標と見なして三辺の長さを計算し表示する。プログラムは3つのファイルQ4.h, Q4m.c, Q4f.cからなり、それぞれヘッダファイル、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、本来の動作を行う実行ファイルが生成される。
Q4m.cとQ4f.cは、それぞれ単体テストが行えるようになっている。Q4m.cのみでコンパイルして実行すると、入出力のテストのみを行い、三角形の実際の長さの計算は行わない。Q4f.cのみでコンパイルして実行すると、キーボード入力無しで固定した初期値を使い、長さの計算のテストを行う。

[全体をコンパイルした場合の実行例]

```
% ./a.out
1 番目の点の座標を入力してください: -4.0 0.0
2 番目の点の座標を入力してください: 0.0 0.0
3 番目の点の座標を入力してください: 0.0 3.0
三角形の辺の長さは  4.000,  3.000,  5.000
%
```

1. プログラムが全体で正しくコンパイル・動作できるように、実行例を参考に最適な語句で空欄(1)～(2)を埋めよ。
2. 単体テストのために、Q4m.c のみでコンパイルし、実行ファイルを作成するときのコンパイルコマンドを解答欄(3)に答えよ。
3. 2.の方法で Q4m.c を単体テスト用にコンパイルしたのち実行し、平面座標として(-4,0), (0,0), (0,3)をこの順に入力した（上の実行例と同じ）。このとき、辺の長さとして表示される内容を解答欄(4)に答えよ。
4. 本来の動作を行う実行ファイルを生成するための手順として、まず Q4m.c と Q4f.c を別々にコンパイルして、実行ファイルの前段階の中間ファイルを作成することができる。この中間ファイルをなんと呼ぶか、解答欄(5)に答えよ。
5. 本来の動作を行う実行ファイルを 4.の手順で作成する際のコマンド群、すなわち Q4m.c と Q4f.c をそれぞれ別々にコンパイルするコマンド、中間ファイルを合わせて実行ファイルを作成するコマンド、全てのコマンドを実行順に解答欄(6)に答えよ。なお、math.h を使用しているため、実行ファイルを作成する段階では -lm オプションが必要である。

[プログラム]

[ファイル Q4.h]

```
typedef struct{
  double x;
  double y;
} XY;

_____(1)_____
_____(2)_____
```

[ファイル Q4m.c]

```
#include <stdio.h>
#include "Q4.h"
#define myab(a)  ((a)<0 ? (-(a)) : (a))
```

次ページに続く

```
int main(){
  XY point[3];
  double dp[3];
  int i;

  for ( i=0; i<3; i++ ){
    printf("%d 番目の点の座標を入力してください: ", i+1);
    scanf("%lf%lf", &point[i].x, &point[i].y);
  }

#ifdef DEBUG
  dp[0] = myab( point[0].x );
  dp[1] = myab( point[1].x );
  dp[2] = myab( point[2].x );
#else
  trilen( point, dp );
#endif

  printf("三角形の辺の長さは  ");
  printf("%.3f,  %.3f,  %.3f\n", dp[0], dp[1], dp[2]);

  return 0;
}
```

[ファイル Q4f.c]

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

#ifdef DEBUG
int main(){
  XY point[3]={{2,0}, {6,0}, {6,3}};
  double dp[3];

  trilen( point, dp );

  printf("三角形の辺の長さは  ");
  printf("%.3f,  %.3f,  %.3f\n", dp[0], dp[1], dp[2]);
  return 0;
}
#endif

void trilen(XY *p, double *dp){
  dp[0] = dist( p[0], p[1] );
  dp[1] = dist( p[1], p[2] );
  dp[2] = dist( p[2], p[0] );
}
```

次ページに続く

```
double dist(XY p1, XY p2){
  double dx, dy;

  dx = p1.x-p2.x;
  dy = p1.y-p2.y;
  return sqrt( dx*dx + dy*dy );
}
```

問題 5 （14 点）

以下のプログラムは、Vector型の構造体または構造体配列に格納された2つの2次元ベクトルの和を3通りの方法で求めている。それぞれの方法は、3つの関数add_Vector1，add_Vector2，add_Vector3で実現されている。各関数の概要は以下の通りである。

1. add_Vector1: 2つの2次元ベクトルを引数として受け取り、和のベクトルを戻り値として返す。
2. add_Vector2: 2つの2次元ベクトルを引数として受け取り、和のベクトルを3つ目の引数として受け取ったVector型の構造体変数へのポインタで示されるアドレスに書き込む。
3. add_Vector3: 要素数3のVector型の構造体配列を受け取り、配列要素0と1に格納されている2つの2次元ベクトルの和を求め、配列要素2に格納する。

実行例とプログラム中のコメントを参考に、最適な語句で空欄を埋め、プログラムを完成させよ。同じ空欄番号には同じ内容が入る。一方、異なる番号の空欄でも同じ内容が入る場合もある。また、空欄(5)〜(7)は複数行にわたる内容を書いてよい。

[実行例]

```
% ./a.out
Result (1): (4.000000, 2.000000)
Result (2): (4.000000, 2.000000)
Result (3): (4.000000, 2.000000)
```

[プログラム]

```
#include <stdio.h>

typedef struct {
  double  x; /* x成分 */
  double  y; /* y成分 */
} Vector;

_____(1)_____ add_Vector1(Vector, Vector);
_____(2)_____ add_Vector2(Vector, Vector, Vector *);
_____(3)_____ add_Vector3(Vector *);

int main()
{
  Vector v1 = {1.0, 0.0}, v2 = {3.0, 2.0};
  Vector va1, va2; /* それぞれ、add_Vector1とadd_Vector2により求めた和のベクトルを格納する */

  Vector v_arr[3]; /* add_Vector3に渡す構造体配列 */
  v_arr[0] = v1;
  v_arr[1] = v2;

  va1 = add_Vector1(v1, v2);
  add_Vector2(v1, v2, _____(4)_____);
  add_Vector3(v_arr);
```

次ページに続く

```
  printf( “Result (1): (%f, %f)\n” , va1.x, va1.y);
  printf( “Result (2): (%f, %f)\n” , va2.x, va2.y);
  printf( “Result (3): (%f, %f)\n” , v_arr[2].x, v_arr[2].y);

  return 0;
}

_____(1)_____ add_Vector1(Vector v1, Vector v2)
{
   Vector va;
```


```
}

_____(2)_____ add_Vector2(Vector v1, Vector v2, Vector *va_ptr)
{
```


```
}

___(3)___ add_Vector3(Vector *v)
{
```


```
}
```

問題 6 （18 点）

ノード中のデータとして文字型のメンバ key を持つ連結リストを取り扱うプログラムを作成したい。
リストへの挿入は、キーボードから大文字のアルファベット 1 文字を入力することで行い、入力された文字のアスキーコードで昇順（小さい順）に並ぶようにリスト内に挿入する。
なお、入力された大文字のアルファベット 1 文字のノードを挿入する場所の特定は、search 関数で行う。search 関数は、引数のアスキーコードよりも大きいアスキーコードの key を持つノードを見つけた場合に、1 つ前のノードを返す。
入力を与えた後は、毎回リストの内容を全て表示する。重複する文字は入力されないものとして、それらのチェックは行わない。また、文字入力の際はアルファベット 1 文字を文字列扱いで配列 data に読み込み、その先頭文字を使用している。
実行例とプログラム中のコメントを参考に、最適な語句で空欄を埋め、プログラムを完成させよ。同じ空欄番号には同じ内容が入る。

[実行例]

```
% ./a.out
大文字のアルファベットを1文字入力してください -> A
Head - A
大文字のアルファベットを1文字入力してください -> Z
Head - A - Z
大文字のアルファベットを1文字入力してください -> G
Head - A - G - Z
大文字のアルファベットを1文字入力してください -> ^D
```

[プログラム]

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

struct node{
        char key; /* 大文字のアルファベット 1 文字 */
        _____(1)_____ next;
};
typedef _____(1)_____ NodePointer;

NodePointer insert(char); /* 新しいノードを挿入 */
NodePointer search(char); /* 新しくノードを挿入する場所を特定 */
void listprint(void); /* リスト内のデータを全て表示 */
NodePointer make_1node(char, NodePointer); /* 新しいノードを作成 */

NodePointer head;

int main(){
    char data[2];
    head = make_1node('\0', NULL); /* head ノードの作成 */
    while(1){
        printf("大文字のアルファベットを 1 文字入力してください -> ");
        if(scanf("%s", data) != 1)break;
        if('A'>data[0] || 'Z'<data[0])break; /* 大文字のアルファベット以外であれば、終了*/
```

次ページに続く

```
        insert(data[0]);
        listprint();
        }
    printf("\n");

    return 0;
}

/* 新しいノードを挿入 */
/* key のアスキーコードが昇順に並ぶように挿入 */
NodePointer insert(char c){
    NodePointer newnode, n;

    n = search(c); /* 新しくノードを挿入する場所を特定 */
    newnode = make_1node(_____(2)_____); /* 新しくノードを作成 */
    _____(3)_____; /* 作成したノードをリストに挿入 */

    return newnode;
}

/* 新しくノードを挿入する場所を特定 */
/*  引数のアスキーコードよりも大きいアスキーコードの key を持つノードを見つけた場合一つ手前のノードを return*/
NodePointer search(char c){
    NodePointer n;

    for(n=head; _____(4)_____; n=n->next){ /* ノードをたどって挿入すべき場所を見つける*/
        if(_____(5)_____ > c)break; /* アスキーコードの比較 */
    }

    return n;
}

/* リスト内のデータを全て表示 */
void listprint(void){
    NodePointer n;

    printf("Head");
    for(n=head->next; _____(6)_____; n=n->next){
        printf(" - %c", _____(7)_____);
    }
    printf("\n");
}
```

次ページに続く

```
/* 新しいノードを作成 */
NodePointer make_1node(char c, NodePointer p){
    NodePointer n;

    if((n = (_____(8)_____)malloc(_____(9)_____)) == NULL){ /* メモリ確保 */
        printf("メモリを確保できない");
        exit(8);
    }

    n->key = c;
    n->next = p;

    return n;
}
```