プログラミングＣ　2025 年度期末試験

2025 年 12 月 1日 13:30- 15:00（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 17 ページ）である。

・途中での提出・退席は 14 時 10 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。

・筆記具と学生証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。

・学生証を忘れた場合は、学生課で仮学生証を発行してもらうこと。
なお、発行のために試験開始時間に遅れた場合、時間延長はありません。

・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。

・実行例ではキーボード入力を太字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（16 点）

テキストファイルから区切り文字（スペース・改行等）で区切られた単語の並びを読み込み、１行の長さが指定した最大文字数を超えないように改行を入れながら、読み込んだ単語をスペース区切りで標準出力に出力するプログラムを作成したい（文字数が指定した数を超えない場合は、もともと改行があった場合でも改行しない。ファイル中の１単語の最大文字数より少ない数が指定されることは考えなくてよい)。ただし、行の先頭にはスペースは出力せず、行末もスペース無しで改行することとする。１行の最大文字数と入力ファイル名は、コマンドライン引数で与える。
実行例やプログラム中のコメントを参考に、空欄にふさわしいコードを答えなさい。

[実行例]

```
% cat in1.data  （元のテキストファイルの内容）
In 1984 Apple Computer introduced the Macintosh
desktop computer with a very "friendly" graphical
user interface. Graphical user interfaces(GUIs) began
to change the complexion of the software industry.
%
% ./a.out 40 in1.data  （一行最大 40 文字として実行）
In 1984 Apple Computer introduced the
Macintosh desktop computer with a very
"friendly" graphical user interface.
Graphical user interfaces(GUIs) began to
change the complexion of the software
industry.
%
% ./a.out 80 in1.data  （一行最大 80 文字として実行）
In 1984 Apple Computer introduced the Macintosh desktop computer with a very
"friendly" graphical user interface. Graphical user interfaces(GUIs) began to
change the complexion of the software industry.
%
```

[プログラム]

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

int main(int argc, char* argv[])
{
  ____(1)____;           /* ファイルポインタ fpin の宣言 */
  char str[128];
  int wlen, llen, lmax; /* 単語の長さ、行の長さ、一行の最大文字数 */

  if (argc < 3){ /* 一行文字数と入力ファイル名の指定がないときは終了 */
    fprintf( stderr, "Usage: ./a.out LineLength InputFile\n" );
    exit (2);
  }
```

次ページに続く

```
  lmax = atoi( argv[1] ); /* 引数で与えた一行の最大文字数 */
  /* 注：atoi 関数は「数字でできた文字列」を、「int 型の整数値」に変換する関数 */

  if ((fpin = ____(2)____) == NULL){ /* 入力ファイルオープン */
    fprintf( stderr, "Cannot open input file!\n");
    exit (3);
  }

  llen = 0;   /* 行の長さを初期化 */
  while ( ____(3)____( fpin, "%s", str ) == 1 ){ /* ファイルから 1 単語読み込み */
    wlen = ____(4)____;   /* 単語の文字数 */
    if (llen == 0){ /* 行の始めであればそのまま出力 */
      llen = wlen;
      fprintf( stdout, "%s", str );
    }
    else {
      llen += ____(5)____; /* スペース 1 つと 1 単語分文字が増える */
      if (____(6)____){ /* 一行文字数が最大文字数以下なら */
       fprintf( stdout, " %s", str );    /* スペースと 1 単語出力 */
      }
      else {
       fprintf( ____(7)____ );    /* 改行してから出力 */
       llen = wlen;    /* 次の行での文字数 */
      }
    }
  }

  fprintf( stdout, "\n" );
  ____(8)____;   /* ファイル操作は終了 */
  return 0;
}
```

問題 2（28 点）

問題２－１

以下のようなプログラムを実行したときの表示を答えなさい。番号は解答用紙に記載済みなので、それ以降の部分を記述すること。以下の説明や%p で表示されるアドレスは便宜上 10 進数で表すものとする。darr の先頭アドレスは 1000 番地、str1 は 2000 番地、str2 は 3000 番地、str3 は 4000 番地とする。書き込み禁止領域は 5000 番地から始まっており、データは最初から余分な隙間無く詰められていると仮定する。なお、変数のサイズなどは会津大学の標準環境に準拠する。

[**プログラム**]

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

int main()
{
  double darr[4]={12.0, 11.1, 7.4, 3.8};
  char *p, str1[]="Aizu";
  char str2[][10]={"Wakamatsu", "Bange", "Misato"};
  char *str3[3]={"Hibarako", "Onogawako", "Akimotoko"};

  printf("(1) %d\n",sizeof(darr));
  printf("(2) %d\n",sizeof(str2));
  printf("(3) %d\n",sizeof(str3));
  printf("(4) %p\n", &darr[3]);
  p=&str1[1];
  printf("(5) %c\n",*p+1);
  p=&str2[1][2];
  printf("(6) %c\n",*(p+2));
  printf("(7) %p\n",p-10);
  p=str3[2];
  printf("(8) %c\n",*p++);
  printf("(9) %s\n",p+3);
  printf("(10) %d\n",strlen(p));

  return 0;
}
```

問題２－２

引数で与えられた文字列を、動的メモリ割り当てを使ってちょうど必要なサイズの文字配列にコピーし、それを戻り値として戻す関数 mystrcpy とその関数の中で使用する文字列の長さを返す関数 mystrlen を作成したい。コメント等を参考に、下線部を埋めて完成させなさい。ただし、文字列操作のライブラリ関数は使用しないものとする。なお、(3)には適切なキャストを入れること。

[プログラム（関数定義部分）]

```
____(1)____mystrcpy(char *s1)
{
   char *s2,*s2org; /* コピー先ポインタ、(s2 は作業用）*/
   int len; /* 文字列の長さ */

   len=mystrlen(___(2)___);

   s2org=s2=(____(3)____)malloc(_________(4)_________);
   while(_____(5)_____) {
     s2++;
     s1++;
   }
   return ___(6)___ ;
}

int mystrlen(____(7)____ *s)
{
  int i;
  for (i=0; _____(8)_____ ; i++) ;

  return ____(9)____;
}
```

問題 3（20 点）

気象観測データを管理するプログラムを作成したい。要件は以下の通りである。
- 観測データを保存する構造体（weather）と、観測所の情報を保存する構造体（station）を作る。
- typedef による名前付けは行わない。
- 構造体 weather は、日付・気温・湿度を保持する。
- 構造体 station は、観測所の名前と、その観測所で観測された複数日の weather データを配列として保持する。
- station 構造体が保持できる観測データの最大数（100）は、マクロ MAX_DATA 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。

＜観測データ：weather＞
・日付（文字型配列：最大 12 文字）: date
・気温（double 型）: temperature
・湿度（int 型）: humidity

＜観測所：station＞
・観測所名（文字型配列：最大 30 文字）: st_name
・観測データ配列（weather 構造体の配列: 最大 100 件）: data
・データ数（int 型）: num_data

このプログラムは以下のように動作する。
- main 関数で station 構造体型変数を宣言する。
- input_data 関数で観測所名と日ごとの観測データをキーボードから入力する。
- 入力は Ctrl+D で終了する。
- print_data 関数で観測所名と観測データを一覧表示する。
- sort_by_temp 関数で気温を基準に降順（高い順）に並べ替える。
- main 関数の最後で、最高気温と最低気温を表示する。

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]

```
% ./a.out
Station Name: Aizu
New Data (Date(YYYY-MM-DD) Temp(°C) Humidity(%)): 2025-11-01 12.3 55
New Data (Date(YYYY-MM-DD) Temp(°C) Humidity(%)): 2025-11-02 18.0 60
New Data (Date(YYYY-MM-DD) Temp(°C) Humidity(%)): 2025-11-03 8.5 70
New Data (Date(YYYY-MM-DD) Temp(°C) Humidity(%)): Ctrl+D
```

次ページに続く

```
====================================
Station Name : Aizu
====================================
Date           Temp(°C)   Humidity(%)
------------------------------------
2025-11-01     12.3       55
2025-11-02     18.0       60
2025-11-03      8.5       70
====================================
Sorting By Temperature
====================================
Station Name : Aizu
====================================
Date           Temp(°C)   Humidity(%)
------------------------------------
2025-11-02     18.0       60
2025-11-01     12.3       55
2025-11-03      8.5       70
====================================
Max:18.0°C   Min:8.5°C
%
```

[**プログラム**]

```
#include <stdio.h>
#include <string.h>
#define MAX_DATA 100

/* 観測データを保存する構造体 */
____(1)____ weather {
    char date[12];
    double temperature;
    int humidity;
};

/* 観測所の情報を保存する構造体 */
____(1)____ station {
    char st_name[30];
    ____(2)____ data[MAX_DATA]; /* weather 型構造体配列 */
    int num_data; /* 読み込んだデータ数 */
};

/* データ入力の関数 */
void input_data(struct station *);

/* 観測データを表示する関数 */
void print_data(struct station *);
```

次ページに続く

```
/* 気温を降順に並べ替える関数 */
void sort_by_temp(struct weather[], int);

int main(){
    ____(3)____ obs; /* station 型構造体変数の宣言 */
    input_data(____(4)____);
    print_data(____(4)____); /* 入力内容の表示 */
    printf("Sorting By Temperature\n");
    sort_by_temp(obs.data, obs.num_data);
    print_data(____(4)____); /* ソート後の表示 */

    /* 最高気温と最低気温の表示 */
    printf("Max:%.1f°C\t", obs.data[0].temperature);
    printf("Min:%.1f°C\n", obs.data[____(5)____].temperature);
    return 0;
}

/* 観測所名、年月日、温度、湿度のデータを入力する関数 */
void input_data(struct station *p){
    int i;
    printf("Station Name: ");
    scanf("%s", ____(6)____);  /* 観測所名を入力 */
    for(i=0;i<MAX_DATA;i++){
        printf("New Data (Date(YYYY-MM-DD) Temp(°C) Humidity(%%)): ");
        if(scanf("%s %lf %d",
                 p->data[i].date,
                 &p->data[i].temperature,
                 ____(7)____) != 3){
            printf("\n");
            break;
        }
    }
    p->num_data = i;
}

/* 観測データを表示する関数 */
void print_data(struct station *p){
    int i;
    printf("====================================\n");
    printf("Station Name : %s\n", p->st_name);
    printf("====================================\n");
    printf("Date           Temp(°C)   Humidity(%%)\n");
    printf("------------------------------------\n");
    for(i=0;i<____(8)____;i++){
        printf("%-12s %8.1f %8d\n",
               p->data[i].date,
               p->data[i].temperature,
               p->data[i].humidity);
```

次ページに続く

```
    }
    printf("=====================================\n");
}

/* 気温を基準に降順に並べ替える関数 */
void sort_by_temp(struct weather data[], int num_data){
    int i,j;
    struct weather tmp;
    for(i=0;i<num_data-1;i++){
        for(j=i+1;j<num_data;j++){
            if(____(9)____ < ____(10)____){
                tmp = data[i];
                data[i] = data[j];
                data[j] = tmp;
            }
        }
    }
}
```

問題 4（20 点）

以下は、三項演算子を用いて、引数として当てられた文字 c が 0-9 までの数字か否かを判定するマクロ関数 isNum の定義である。c が数字の場合は TRUE を、また数字以外の文字である場合は FALSE を返す（TRUE と FALSE は別途マクロとして定義されているものとし、それぞれ 1, 0 の値を持つ）。空欄を埋めてマクロ関数として完成させよ。

___(1)___ isNum(c) ((___(2)___ && ___(3)___) ? ___(4)___)

また以下は、マクロ関数 isNum を用いてキーボードから入力された文字が数字か否かを判定し、結果を表示するプログラムである。マクロ isNum、TRUE、FALSE はヘッダファイル Q4.h の中で定義されている。プログラムのファイル名は Q4.c であり、Q4.h と Q4.c は共にカレントディレクトリにあるものとする。このプログラムは、表示内容を英語と日本語に切り替えることができるようになっている。この切り替えはコンパイル時にマクロ ENGLISH が定義されているか否かで行われ、定義されている場合は英語表示、そうでない場合は日本語表示をする実行ファイルが生成される。
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。

[プログラム]

Q4.c（Q4.h の内容は省略）

```
#include <stdio.h>
___(5)___

int main(){
    char c;

    ___(6)___ ENGLISH
    printf("Input any character: ");
    ___(7)___
    printf("何か文字を入力してください: ");
    ___(8)___
    scanf("%c", &c);

    if (___(9)___) {
        ___(6)___ ENGLISH
        printf("%c is a number.\n", c);
        ___(7)___
        printf("%c は数字です。\n", c);
        ___(8)___
    } else {
        ___(6)___ ENGLISH
        printf("%c is not a number.\n", c);
        ___(7)___
        printf("%c は数字ではありません。\n", c);
        ___(8)___
    }
}
```

次ページに続く

プログラム Q4.c を英語表示モードでコンパイルするためのコマンドを書け。

___(10)___

問題５（20 点）

任意の正の整数 n に対し、「偶数なら n/2、奇数なら 3n+1 を新しい n とする」という操作を繰り返すと、必ず 1 に到達するという「コラッツ予想」は、数学の未解決問題として有名です。

例えば n=6 の場合、

6 (偶) -> 3 (奇) -> 10 (偶) -> 5 (奇) -> 16 (偶) -> 8 (偶) -> 4 (偶) -> 2 (偶) -> 1

となり、8 回のステップで 1 に到達します。

この「1 に到達するまでのステップ数」を計算する再帰関数 collatzSteps を実装し、プログラムを完成させるため、【プログラム】の (1) から (10) に入る適切な字句や識別子、型名を答えなさい。

各関数の仕様

1. **collatzSteps 関数**
   - 関数名: collatzSteps;
   - 処理: 再帰を用いて、整数 n が 1 に到達するまでのステップ数を返す。
     - 基底ケース: n が 1 の場合、ステップ数は 0 なので 0 を返す。
     - 再帰ステップ（偶数）: n が偶数の場合、「n/2 のステップ数」に 1 を加えた値を返す。
     - 再帰ステップ（奇数）: n が奇数の場合、「3n + 1 のステップ数」に 1 を加えた値を返す。
2. **main 関数**
   - 1 から 10 までの各数について collatzSteps 関数を呼び出し、結果を表示する。

期待される出力

```
Steps for 1: 0
Steps for 2: 1
Steps for 3: 7
Steps for 4: 2
Steps for 5: 5
Steps for 6: 8
Steps for 7: 16
Steps for 8: 3
Steps for 9: 19
Steps for 10: 6
```

[プログラム]

```
#include <stdio.h>
/* --- 関数のプロトタイプ宣言 --- */
____(1)____ collatzSteps (____(4)____);


/* --- main 関数 --- */
int main(void) {
    int i, steps;
    /* 1 から 10 までの数で試す */
    for (i = 1; _____(2)_____ ; i++) {
```

次ページに続く

```
        /* 関数を呼び出してステップ数を取得 */
        steps = _____(3)_____;
        printf("Steps for %d: %d\n", i, steps);
    }

    return 0;
}



/* --- 関数定義: collatzSteps --- */
/* n が 1 に到達するまでのステップ数を「再帰」で計算する */
____(1)____ collatzSteps(____(4)____ n) { /* 引数の型*/

    /* 基底ケースの処理 */
    if (_____(5)_____) {
        return _____(6)_____;
    }

    /* (再帰ステップ) n が偶数か奇数か */
    if (_____(7)_____) {
        /* 偶数の場合 */
        return _____(8)_____ + collatzSteps(_____(9)_____);
    } else {
        /* 奇数の場合 */
        return _____(8)_____ + collatzSteps(_____(10)_____);
    }
}
```

問題６（16 点）

講義第９回の連結リストのプログラムは、データとして整数１つを各ノードに格納できるものだった（ノード構造体のメンバーは、int 型変数１つと、次のノードを示すポインタになっていた)。この問題では、名前と点数のデータを持つ構造体（NSDATA 型と名付ける）をデータとしてノードに格納するように、講義で扱った連結リストを改造したい。

[仕様の変更点]

・ NSDATA 型構造体のメンバーは、name（要素数 20 の char 型配列)、score（int 型変数）とする。
・ 挿入関数 insert() はデータの入った NSDATA 型構造体を引数とする。
・ ノード作成関数 make_1node() は、NSDATA 型構造体と、次のノードのアドレスを引数とする。
・ 表示関数 listprint() は、リストをたどって名前と点数を表示する。なお、head ノードには意味のあるデータは入っていないので、表示は head の次のノードから行う。
・ 検索関数 finditem() は、検索したい名前を引数とする。一致する名前が見つかればその一つ前のノードを戻り値とし、見つからなければ NULL をリターンする。
・ 最初にファイルから（名前・点数）データを読み込んでリストを作成し、その後キーボードから入力したデータを追加する。
・ 簡単のため、ヘッダーファイル list.h の内容はプログラムのファイルに含めて書いた。

上記の仕様を満たすように、プログラム中の空欄に入る適切なコードを答えなさい。なお、同じ番号の空欄には同じものが入る。

[実行例]

```
% ./a.out
 [tanaka:39] [sano:63] [kato:92] [aizawa:75]   （これはファイルから読み込んだデータ）
Insert (enter name and score):
nakano 50          （キーボードからデータ入力）
 [nakano:50] [tanaka:39] [sano:63] [kato:92] [aizawa:75]
Insert (enter name and score):
hashimoto 99       （キーボードからデータ入力）
 [hashimoto:99] [nakano:50] [tanaka:39] [sano:63] [kato:92] [aizawa:75]
Insert (enter name and score):
Ctrl+D
%
```

[プログラム]

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

/* 構造体の宣言 */
typedef struct {
  char name[20];
  int score;
}NSDATA;
```

次ページに続く

```
struct node {
  NSDATA data;
  struct node *next;
};

typedef struct node * NodePointer;

/* 関数のプロトタイプ宣言 */
NodePointer insert(NSDATA);
NodePointer finditem(_____(1)_____);
void         listprint(void);
NodePointer make_1node(NSDATA, NodePointer);

/* 外部変数として head を宣言する */
NodePointer head;

int main(){
  NSDATA dt;   /* データ入力用の構造体 */
  FILE *fp;

  /* ファイルポインタ fp を使い、入力ファイルのオープンと
  エラー処理を行う（都合により内容は省略） */

  head = make_1node( dt, NULL );   /* head ノードの作成 */

  /* ファイルから名前と点数を読み込んでリストに挿入
  （都合により内容は省略）*/

  listprint();   /* まずリストの内容を表示する */

  while (1){
    printf("Insert (enter name and score):\n");
    if(scanf("%s%d", _____(2)_____) != 2) break; /* キーボードから追加入力 */
    /* 挿入関数を呼ぶ。同じ名前が既に存在した場合はメッセージを表示 */
    if (insert( dt ) == NULL) printf("%s is already on the list\n", dt.name);
    else listprint();   /* ノードを追加したら、リストの内容を表示 */
  }
  /* メモリ解放は省略 */
  return 0;
}

/* データを引数に与えて、新しいノードを挿入する関数 */
NodePointer insert(NSDATA d)
{
  NodePointer newnode;

  if(finditem( _____(3)_____ ) == NULL){  /* まず名前が既にリストにあるか検索 */
```

次ページに続く

```
    /* 新しいノードを作成し、head の直後に挿入する */
    newnode = make_1node(_____(4)_____);
    head->next = newnode;

    return newnode;
  }
  else return NULL; /* 既に存在した場合は NULL を返す */
}

/* リストの内容を表示する関数 */
void listprint(void)
{
  NodePointer n;
  for(n = _____(5)_____; _____(6)_____; n = n->next) {
    printf( " [%s:%d]", n->data.name, n->data.score );
  }
  printf( "\n" );
}

/* 名前が一致するノードを検索する関数 */
NodePointer finditem(_____(1)_____keyname)
{
  NodePointer n;
  for(n = head; n->next != NULL; n = n->next){
    if( strcmp( _____(7)_____, keyname ) == 0) return n; /* 引数とデータを比較 */
  }
  return NULL; /* 見つからなかった時は NULL をリターン */
}

/* 新しいノードを作成する関数 */
NodePointer make_1node(NSDATA d, NodePointer p)
{
  NodePointer n;
  /* ノード一つ分のメモリ確保 */
  if((n = (NodePointer)malloc(_____(8)_____)) == NULL) {
    printf("Error in memory allocation\n");
    exit(8);
  }
  n->data = d;   /* データをノードに格納 */
  n->next = p;   /* 次のノードとつなぐ */
  return n;   /* 新しいノードのアドレスをリターン */
}
```