情報理工学演習III　演習課題（第2回）

テーマ　　：基礎的なデータ構造
実施日　　：2020年4月23日（木）
担当　　：有村 博紀，堀山 貴史，金森 憲太朗（TF），加井 丈志（TA）
授業連絡先：enshuiii-csit@ist.hokudai.ac.jp（演習課題の提出・質問など）
担当連絡先：kanamori@ist.hokudai.ac.jp（金森），kai@ist.hokudai.ac.jp（加井）

以下の各問に関して，それぞれプログラムを実装せよ． 実装したコードは必ず，以下に示す提出様式を守って，**本授業のメールアドレス（enshuiii-csit@ist.hokudai.ac.jp）宛にメールで提出**すること．

- メールの件名　：「[csit ex3]xxxxxxxx」，ここで xxxxxxxx は自分の学生番号．
- 添付ファイル名：第 n 回演習の問 m に対するコードのファイル名は，「n_m.c」とする．

例：学生番号02180000の学生の場合

- メール宛先　：enshuiii-csit@ist.hokudai.ac.jp
- メール件名　：[csit ex3]02180000
- 添付ファイル：2_1.c ，2_2.c ，2_3.c ，2_4.c

メール本文には，学生番号，所属コース，名前，実施回（第2回演習），解いた問題番号を記載すること． 課題提出によって出席をとるため，解いている途中の状態でも構わないので，演習時間内に一度は提出すること．

実装にはC言語を用い，gcc（ver.4.8.5 以降）でコンパイルすること．

コンパイル・実行例

```
% gcc 2_1.c        // コンパイル
% ./a.out          // 実行
```

演習時間内に解き終わらなかった場合は，次回の演習時間までに提出すること．期間中であれば，何度でも再提出してよい． 提出状況等は本授業のホームページまたはELMSグループ（moodle）上に掲示予定なので，適宜確認すること．

問題番号に * が付されたものは，発展課題である．解答は必須ではないが，解ければ成績評価に加点されるので，是非挑戦してほしい．

【問1】 以下の Sample 1 は，

1. int 型の値を格納する空のリスト L を作成
2. L の先頭に要素を 5，3，5，6 とするセルを順に追加し，L を表示
3. L から要素が 5 のセルをすべて削除し，先頭に 5 を要素とするセル追加して，L を表示

を行うプログラムである． これを 2_1.c として実装せよ．

Sample 1

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: typedef struct _cell {
5:     int element;
```

```
 6:     struct _cell *next;
 7: }cell;
 8:
 9: typedef struct _list{
10:     cell *head;
11: }list;
12:
13: /* 空のリストを作成する関数 */
14: list* create() {
15:     list *L = (list *)malloc(sizeof(list));
16:     L->head = (cell *)malloc(sizeof(cell));
17:     L->head->next = NULL;
18:     L->head->element = -1;
19:     return L;
20: }
21:
22: /* リスト L の先頭に,要素 element を持つセルを
23:    定数時間で追加する関数 */
24: void addFirst(list *L, int element) {
25:     cell *add = (cell *)malloc(sizeof(cell));
26:     add->element = element;
27:     add->next = L->head->next;
28:     L->head->next = add;
29: }
30:
31: /* リスト L から,要素 element を持つすべてのセルを
32:    線形時間で削除する関数 */
33: void del(list *L, int element) {
34:     cell *ptr    = L->head->next;
35:     cell *prev = L->head;
36:     while(ptr != NULL){
37:         if(ptr->element == element) {
38:             prev->next = ptr->next;
39:         } else {
40:             prev = ptr;
41:         }
42:         ptr    = ptr->next;
43:     }
44: }
45:
46: /* リスト L を先頭から順に表示する関数 */
47: void print(list *L) {
48:     cell *ptr = L->head->next;
49:     while(ptr != NULL) {
50:         printf("%d ", ptr->element);
51:         ptr = ptr->next;
52:     }
53:     printf("\n");
54: }
55:
56: int main() {
```

```
57:    list *L = create();
58:
59:    addFirst(L, 5);
60:    addFirst(L, 3);
61:    addFirst(L, 5);
62:    addFirst(L, 6);
63:    print(L);
64:
65:    del(L, 5);
66:    addFirst(L, 5);
67:    print(L);
68:
69:    return 0;
70: }
```

実装したプログラム 2_1.c を実行し，結果を確認せよ．

【問2】 以下を実行するプログラムを 2_2.c として実装せよ．

1. int 型の値を格納する空のスタック S を作成
2. S に 5，3，6 を順に push し，S を表示
3. S の先頭の要素を pop した後，3，7 を順に push し，S を表示

このとき，Sample 1 を参考に，以下のようにそれぞれ関数を設計すること．

- stack* create()：空のスタックを作成する関数
- void push(stack *S, int element)：S に要素 element を定数時間で push する関数
- void pop(stack *S)：S の先頭要素を定数時間で pop する関数
- void print(stack *S)：S を先頭から順に表示する関数

また，スタックの表示は Sample 1 中の print 関数の出力と同様に，各要素の直後に半角スペースを置いてこれらを区切り，スタック全体の出力の末尾には改行を入れること．

実装したプログラム 2_2.c を実行し，結果を確認せよ．

（注）スタックは，「後入れ先出し」のデータ構造である．一般に，スタックの先頭に要素を格納することを push といい，スタックの先頭の要素を削除（取り出し）することを pop という．

【問3】 以下を実行するプログラムを 2_3.c として実装せよ．

1. int 型の値を格納する空のキュー Q を作成
2. Q に 5，3，6 を順に enqueue し，Q を表示
3. Q の先頭の要素を dequeue した後，3，7 を順に enqueue し，Q を表示

このとき，Sample 1 を参考に，以下のようにそれぞれ関数を設計すること．

- queue* create()：空のキューを作成する関数
- void enqueue(queue *Q, int element)：Q に要素 element を定数時間で enqueue する関数
- void dequeue(queue *Q)：Q の先頭要素を定数時間で dequeue する関数
- void print(queue *Q)：Q を先頭から順に表示する関数

また，キューの表示は Sample 1 中の print 関数の出力と同様に，各要素の直後に半角スペースを置いてこれらを区切り，キュー全体の出力の末尾には改行を入れること．

実装したプログラム 2_3.c を実行し，結果を確認せよ．

（注）キューは，「先入れ先出し」のデータ構造である．一般に，キューの末尾に要素を格納することを enqueue といい，キューの先頭の要素を削除（取り出し）することを dequeue という．

【問4*】 逆ポーランド記法とは，演算子を被演算子の直後に配置する数式の記述方法である．たとえば，$(1+2)*(3+4)$ という数式は，逆ポーランド記法では $1\,2+3\,4+*$ と記述される．

逆ポーランド記法で記述された四則演算を含む数式を受け取り，その計算結果を出力するプログラムを 2_4.c として実装せよ（ヒント：スタックを用いるとよい）．このとき，数式中に含まれる数値はすべて非負整数であり，計算途中における各演算の評価結果はすべて非負整数であることが保証されるものとする（要するに，int 型の要素を格納するスタックを用いるとよい）．
入力は以下の形式で標準入力から与えられる．数式の各項はアンダーバー "_" で区切られているものとし，数式は文字数にして100文字以内であるとしてよい．出力の末尾には改行を入れること．

入力形式
数式

入力例
1_2_+_3_4_+_*

実装したプログラム 2_4.c に，標準入力から 10_3_*_7_*_2_/_9_3_/_5_2_-_*_+ を入力して実行し，結果を確認せよ．