課題3

ソースコード

作成したコードは下記のようになった

```
/*
入力：通常の数式(今回は文字式のため数字の関連で分岐をする必要はない
出力：入力を逆ポーランド記法にしたもの
*/
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef char data_type;
typedef struct node_tag {
        data_type data;
        struct node_tag *next;
} node_type;

void initialize(node_type ** pp)
{
        *pp = NULL;
}

int ret_rank(char ch)
{
        if (ch == '=') {
                return 0;
        } else if (ch == ')') {
                return 1;
        } else if ((ch == '+') || (ch == '-')) {
                return 2;
        } else if ((ch == '/') || (ch == '*')) {
                return 3;
        } else if (ch == '(') {
                return 4;
        } else {
                return 5;
        }
}

int is_empty(node_type * p)
{
        if (p == NULL) {
                return true;
        } else {
                return false;
        }
```

```
}

data_type top(node_type * p)
{
        if (p == NULL) {
                return '\0';
        } else {
                return p->data;
        }
}

node_type *new_node(data_type x, node_type * p)
{
        node_type *temp;
        temp = (node_type *) malloc(sizeof(node_type));
        if (temp == NULL) {
                return NULL;
        } else {
                temp->data = x;
                temp->next = p;
                return temp;
        }
}

int push(data_type x, node_type ** pp)
{
        node_type *temp;
        temp = new_node(x, *pp);
        if (temp == NULL) {
                return false;
        }
        *pp = temp;
        return true;
}

int pop(node_type ** pp)
{
        node_type *temp;
        if (*pp != NULL) {
                temp = (*pp)->next;
                free(*pp);
                *pp = temp;
                return true;
        } else {
                return false;
        }
}

void reverse_polish()
{
        char ch;
        char read_str[128];
        char out_ch;
        int i;
```

```
        node_type *stack;
        initialize(&stack);

        i = 0;

        while ((ch = getchar()) != EOF) {
                read_str[i++] = ch;
                if (i >= 128 - 1) {
                        break;
                }
                if(ch == ' ' || ch == '\n' ||ch == '\0') continue;
                while (!is_empty(stack) && (ret_rank(top(stack)) != 4)
                       && (ret_rank(ch) <= ret_rank(top(stack)))) {
                        out_ch = top(stack);
                        if (out_ch != '\n' && out_ch != '\0' && out_ch != EOF) {
                                printf("%c", top(stack));
                        }
                        pop(&stack);
                }
                if (ret_rank(ch) != 1) {
                        push(ch, &stack);
                } else {
                        pop(&stack);
                }
        }
        while (is_empty(stack) == false) {
                out_ch = top(stack);
                if (out_ch != '\n' && out_ch != '\0' && out_ch != EOF) {
                        printf("%c", top(stack));
                }
                pop(&stack);
        }
        printf("%c", read_str[i - 1]);
        return;
}

int main(void)
{
        reverse_polish();
        return 0;

}
```

実行結果

入力例は次のとおりである

in3-1.txt

```
A = ( B + C ) * D - E / F
```

in3-2.txt

```
A = ( B + C * D - E ) / F
```

in3-3.txt

```
A = B + C * ( D - E / F )
```

実行結果は次の通りである

```
$ gcc -Wall 3.c
$ ./a.out < in3-1.txt
ABC+D*EF/-=
$ ./a.out < in3-2.txt
ABCD*+E-F/=
$ ./a.out < in3-3.txt
ABCDEF/-*+=
```

課題4

ソースコード

作成したコードは下記のようになった

```
/*
入力：整数(int型)
出力：入力された数値群を昇順に出力
** 二分探索木を使う **
DELというマクロが使われている場合は課題4の発展問題である削除関数の動作を確認するため
*/
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/*
「課題4に対するヒント」というタイトルのPDFがあるのでそれに合わせた二分探索木を作る
*/
```

```
typedef int data_type;
typedef int freq_type;
typedef struct node_tag {
        data_type data;
        freq_type freq;
        struct node_tag *left;
        struct node_tag *right;
} node_type;

void output(node_type * p)
{                                   //前順で表示
        if (p != NULL) {
                output(p->left);
                for (int i = p->freq; i > 0; i--) {
                        printf("%d ", p->data);
                }
                output(p->right);
        }
}

void initialize(node_type ** pp)
{                                   //初期化関数
        *pp = NULL;
}

int is_member(data_type x, node_type * p)
{                                   //xと同じデータが木の中に存在するか(返値は true,
false )
        if (p == NULL)
                return false;
        else {
                if (x == p->data)
                        return true;
                else {
                        if (x < p->data)
                                return is_member(x, p->left);
                        else
                                return is_member(x, p->right);
                }
        }
}

node_type *min(node_type **p)
{                                   //木の中の最小値を取得
        node_type *min = NULL;

        while ((*p)->left != NULL) {
                p = &(*p)->left;
        }
        min = *p;
        return min;
}

node_type *max(node_type * p)
```

```
{
        while (p->right != NULL) {
                p = p->right;
        }
        return p;
}

node_type *new_node(data_type x)
{                                   //xの入った空の木を作成
        node_type *temp;

        temp = (node_type *) malloc(sizeof(node_type));

        if (temp == NULL)
                return NULL;
        else {
                temp->data = x;
                temp->left = NULL;
                temp->right = NULL;
                temp->freq = 1;
                return temp;
        }
}

int insert(data_type x, node_type ** pp)
{                                   //木へ新しいデータを追加
        node_type *temp;
        if (*pp == NULL) {
                temp = new_node(x);
                if (temp == NULL)
                        return false;
                *pp = temp;
                return true;
        } else if (x == (*pp)->data) {
                (*pp)->freq = (*pp)->freq + 1;
        } else {
                if (x < (*pp)->data)
                        insert(x, &((*pp)->left));
                else if (x > (*pp)->data)
                        insert(x, &((*pp)->right));
        }
}

node_type **find_key(int key, node_type **root)
{
        node_type **node = root;
        while(*node != NULL){
                if(key == (*node)->data)
                        break;
                else if(key < (*node)->data)
                        node = &(*node)->left;
                else if(key > (*node)->data)
                        node = &(*node)->right;
        }
```

```
        return node;
}

int dell_node(int num, node_type ** tree)
{
        node_type *target = NULL;
        node_type **node = NULL;
        node = find_key(num, tree);

        target = *node;
        if(target->freq > 1){
                target->freq--;
                return true;
        }
        if(target->right == NULL)
        {
                *node = target->left;
        }
        else if(target->right->left == NULL)
        {
                *node = target->right;
                (*node)->left = target->left;
                (*node)->right = target->right->right;
        }
        else{
                *node = min(&target->right);
                (*node)->right = target->right;
                (*node)->left = target->left;
        }
        return true;
}

node_type *make_tree(void)
{
        char buf[32];
        int point;
        node_type *root;
        char type;
        initialize(&root);
        fgets(buf, sizeof(buf), stdin);
        sscanf(buf, "%c %d",&type, &point);
        root = new_node(point);

        while (fgets(buf, sizeof(buf), stdin) != NULL) {
                sscanf(buf, "%c %d",&type, &point);

                if(type == 'i'){
                        insert(point, &root);
                }
                else if(type == 'd'){
                        dell_node(point, &root);
                }
        output(root);
        printf("\n");
```

```
        }
        fflush(stdin);
        return root;
}

int main(void)
{
        make_tree();
        return 0;
}
```

実行結果

入力例は次の通りである

in4.txt

```
i 4
i 5
i 6
i 7
i 1
i 4
i 5
i 6
i 7
i 14
i 5
i 6
i 7
i 1
d 1
d 1
d 14
d 4
d 4
d 5
d 5
d 5
d 6
d 6
d 6
d 7
d 7
d 7
```

実行結果は次のようになった

```
$ gcc -Wall 4.c
4.c: In function 'insert':
4.c:109:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
$ ./a.out < in4.txt
4 5
4 5 6
4 5 6 7
1 4 5 6 7
1 4 4 5 6 7
1 4 4 5 5 6 7
1 4 4 5 5 6 6 7
1 4 4 5 5 6 6 7 7
1 4 4 5 5 6 6 7 7 14
1 4 4 5 5 5 6 6 7 7 14
1 4 4 5 5 5 6 6 6 7 7 14
1 4 4 5 5 5 6 6 6 7 7 7 14
1 1 4 4 5 5 5 6 6 6 7 7 7 14
1 4 4 5 5 5 6 6 6 7 7 7 14
4 4 5 5 5 6 6 6 7 7 7 14
4 4 5 5 5 6 6 6 7 7 7
4 5 5 5 6 6 6 7 7 7
5 5 5 6 6 6 7 7 7
5 5 6 6 6 7 7 7
5 6 6 6 7 7 7
6 6 6 7 7 7
6 6 7 7 7
6 7 7 7
7 7 7
7 7
7
```