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

#define MAX_TERMS 101
#define MAX_VARIABLES 2

struct Term{
        float coef;
        char variable[MAX_VARIABLES];
        int expon[MAX_VARIABLES];
};
struct Polynomial{
        struct Term terms[MAX_TERMS];
        int num_terms;
};

void printEach_polynomial(struct Polynomial poly){
        int i;
        int term_printed = 0;

        for(i = 0; i < poly.num_terms; ++i){
                if(poly.terms[i].coef != 0){
                        if(poly.terms[i].coef > 0 && term_printed){
                                printf(" + ");
                        }
                        if(poly.terms[i].variable[0] != 0 &&
poly.terms[i].variable[0] > 0){
                                if(poly.terms[i].coef > 0){
                                        printf("%.2f", poly.terms[i].coef);
                                }
                                else if(poly.terms[i].coef < 0){
                                        printf(" - ");
                                        printf("%.2f", -poly.terms[i].coef);
                                }
                                for (int j = 0; j < MAX_VARIABLES; ++j){
                                        if(poly.terms[i].variable[j] != 0){
                                                printf("%c",
poly.terms[i].variable[j]);

                                                if (poly.terms[i].expon[j] > 1){
                                                        printf("^%d",
poly.terms[i].expon[j]);
                                                }
                                        }
                                }
                        }
                        else{
                                printf(" %.2f", poly.terms[i].coef);
                                if (poly.terms[i].variable[1] != 0){
                                        printf("%c^%d",
poly.terms[i].variable[1], poly.terms[i].expon[1]);
                                }
                        }
                        term_printed = 1;
                }
        }
        if (!term_printed){
                ;
        }
        printf("\n");
}
struct Polynomial mat(struct Polynomial A, struct Polynomial B){
        struct Polynomial result;
        result.num_terms = 0;
        for (int i = 0; i < A.num_terms; ++i){
                for(int j = 0; j < B.num_terms; ++j){
                        if (A.terms[i].variable[0] == B.terms[j].variable[0]){
                                result.terms[result.num_terms].coef =
A.terms[i].coef * B.terms[j].coef;
                                result.terms[result.num_terms].variable[0] =
A.terms[i].variable[0];
                                result.terms[result.num_terms].expon[0] =
A.terms[i].expon[0] + B.terms[j].expon[0];
                                result.terms[result.num_terms].variable[1] = 0;
                                result.terms[result.num_terms].expon[1] = 0;
                                result.num_terms++;
                        }
                        else{
                                result.terms[result.num_terms].coef =
A.terms[i].coef * B.terms[j].coef;
                                result.terms[result.num_terms].variable[0] =
A.terms[i].variable[0];
                                result.terms[result.num_terms].expon[0] =
A.terms[i].expon[0];
                                result.terms[result.num_terms].variable[1] =
B.terms[j].variable[0];
                                result.terms[result.num_terms].expon[1] =
B.terms[j].expon[0];
                                result.num_terms++;
                        }
                }
        }
        return result;
}
struct Polynomial yx2xy(struct Polynomial A){
        int exist_variable_cnt = 0;
        for (int i = 0; i < A.num_terms; ++i){
                if (A.terms[i].variable[0] >= 'a' && A.terms[i].variable[0] <= 'z'
){
                        exist_variable_cnt++;
                }
                if(A.terms[i].variable[1] >= 'a' && A.terms[i].variable[1] <= 'z'
){
                        exist_variable_cnt++;
                }
                if (exist_variable_cnt == 2){
                        if(A.terms[i].variable[0] == 'y'){
                                char temp = A.terms[i].variable[0];
                                A.terms[i].variable[0] = A.terms[i].variable[1];
                                A.terms[i].variable[1] = temp;

                                int temp2 = A.terms[i].expon[0];
                                A.terms[i].expon[0] = A.terms[i].expon[1];
                                A.terms[i].expon[1] = temp2;
                        }

                }
                exist_variable_cnt = 0;
        }
        char temp;
        int temp2;
        for (int i = 0; i < A.num_terms; ++i){
                if (A.terms[i].variable[0] == '\0' && A.terms[i].variable[1] !=
'\0'){
                        temp = A.terms[i].variable[1];
                        A.terms[i].variable[1] = A.terms[i].variable[0];
                        A.terms[i].variable[0] = temp;

                        temp2 = A.terms[i].expon[1];
                        A.terms[i].expon[1] = A.terms[i].expon[0];
                        A.terms[i].expon[0] = temp2;
                }
        }
```

```
  120:         return A;
  121: }
  122: int compare(const void* a, const void* b){
  123:         const struct Term* term1 = (const struct Term*)a;
  124:         const struct Term* term2 = (const struct Term*)b;
  125:
  126:         if (term1->variable[0] == 'x' && term2->variable[0] == 'x'){
  127:                 if (term1->expon[0] == term2->expon[0]){
  128:                         if(term1->variable[1] == 'y' && term2->variable[1] != 'y'
){
  129:                                 return -1;
  130:                         }
  131:                         else if(term1->variable[1] != 'y' && term2->variable[1]
== 'y'){
  132:                                 return 1;
  133:                         }
  134:                 }
  135:         }
  136:         else if(term1->variable[0] != 'x' && term2 ->variable[0] == 'x'){
  137:                 return 1;
  138:         }
  139:
  140:         if(term1 -> variable[0] == 'x' && term2 ->variable[0] != 'x'){
  141:                 return -1;
  142:         }
  143:         else if(term1->variable[0] != 'y' && term2 ->variable[0] == 'y'){
  144:                 return 1;
  145:         }
  146:
  147:         if (term1->variable[0] == term2->variable[0]){
  148:                 if(term1->variable[1] == 'y' && term2->variable[1] == '\0'){
  149:                         return 1;
  150:                 }
  151:                 else if(term1->variable[1] == '\0' && term2->variable[1] == 'y'){
  152:                         return -1;
  153:                 }
  154:
  155:                 if (term1->expon[0] != term2->expon[0]){
  156:                         return term2->expon[0] - term1->expon[0];
  157:                 }
  158:
  159:                 if (term1->variable[1] == 'y' && term2 -> variable[1] == 'y'){
  160:                         return term2->expon[1] - term1->expon[1];
  161:                 }
  162:         }
  163:         return 0;
  164: }
  165:
  166: struct Polynomial simplify(struct Polynomial result){
  167:         for(int i = 0; i < result.num_terms; ++i){
  168:                 for(int j = i + 1; j < result.num_terms; ++j){
  169:                         if(result.terms[i].variable[0] ==
result.terms[j].variable[0] &&
  170:                                         result.terms[i].variable[1] ==
result.terms[j].variable[1] &&
  171:                                         result.terms[i].expon[0] ==
result.terms[j].expon[0] &&
  172:                                         result.terms[i].expon[1] ==
result.terms[j].expon[1]) {
  173:                                 result.terms[i].coef += result.terms[j].coef;
  174:                                 result.terms[j].coef = 0;
  175:                         }
  176:                 }
  177:         }
  178:         int newIndex = 0;
  179:         for (int i = 0; i < result.num_terms; ++i){
  180:                 if(result.terms[i].coef != 0){
  181:                         result.terms[newIndex] = result.terms[i];
  182:                         newIndex++;
  183:                 }
  184:         }
  185:         result.num_terms = newIndex;
  186:
  187:         return result;
  188: }
  189:
  190: int main(){
  191:         struct Polynomial A;
  192:         struct Polynomial B;
  193:
  194:         A.num_terms = 3;
  195:         A.terms[0].coef = 3;
  196:         A.terms[0].variable[0] = 'y';
  197:         A.terms[0].expon[0] = 7;
  198:
  199:         A.terms[1].coef = 4;
  200:         A.terms[1].variable[0] = 'x';
  201:         A.terms[1].expon[0] = 4;
  202:
  203:         A.terms[2].coef = -1;
  204:         A.terms[2].variable[0] = 0;
  205:         A.terms[2].expon[0] = 0;
  206:
  207:         B.num_terms = 3;
  208:         B.terms[0].coef = 5;
  209:         B.terms[0].variable[0] = 'x';
  210:         B.terms[0].expon[0] = 4;
  211:
  212:         B.terms[1].coef = -3;
  213:         B.terms[1].variable[0] = 'y';
  214:         B.terms[1].expon[0] = 2;
  215:
  216:         B.terms[2].coef = 7;
  217:         B.terms[2].variable[0] = 0;
  218:         B.terms[2].expon[0] = 0;
  219:
  220:         struct Polynomial result = mat(A, B);
  221:         struct Polynomial yx2xyresult = yx2xy(result);
  222:         qsort(yx2xyresult.terms, yx2xyresult.num_terms, sizeof(struct Term),
compare);
  223:         struct Polynomial simplifiedResult = simplify(yx2xyresult);
  224:         printEach_polynomial(simplifiedResult);
  225:
  226:         return 0;
  227: }
  228:
```