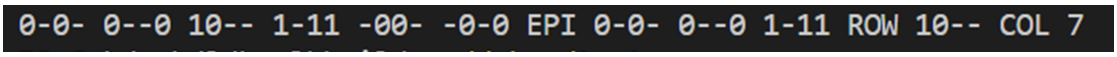
결과보고서

** 해당 과제는 1차, 2차 과제의 코드를 수정하여 구현한 것으로, row & column dominance 동작 위주로 설명합니다.

```
vector<vector<string>> init(int bitnum, int mintermnum, vector<int> minterm);
void deduplicateStr(vector<string>& str);
void deduplicateInt(vector<int>& integer);
void findPI(vector<vector<string>> bit, int bitnum, vector<string>& answer);
void findEPI(vector<vector<string>> bit, vector<string>& answer,
  vector<vector<int>>& epi, vector<string>& exceptepi);
void findRow(vector<vector<int>> epi, vector<string> exceptepi, vector<string>& answer);
void findCol(vector<vector<int>> epi, vector<string> exceptepi,
  vector<string>& answer, vector<int> minterm);
int main();
```

- main
  - `main`에서는 변수의 개수, minterm의 개수, minterm에 대해 입력을 받고 있습니다. -
  - 입력받은 minterm을 2진수 배열로 바꾸는 함수 `init`을 실행하고 `findPI` → `findEPI` → `findRow` → `findCol` 순으로 함수를 실행시킵니다.
  - 각각의 함수에 대한 결과는 스트링 배열 answer에 합해져 출력됩니다.
  - 출력 결과는 다음과 같은 형식으로 나타납니다.

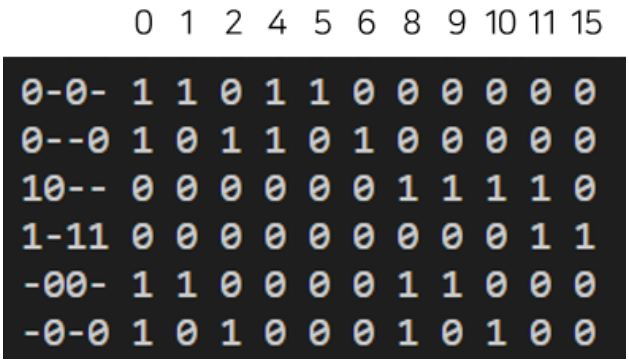



```
int main() {
    int bitnum, mintermnum, min;
    vector<int> minterm;
    vector<string> answer;
    cin >> bitnum >> mintermnum;
    for (int i = 0; i < mintermnum; i++) {
        cin >> min;
        minterm.push_back(min);
    }

    vector<vector<string>> bit = init(bitnum, mintermnum, minterm);
    findPI(bit, bitnum, answer);
    vector<int> v(mintermnum, 0);
    vector<vector<int>> epi(answer.size(), v);
    vector<string> exceptepi = answer;
    findEPI(bit, answer, epi, exceptepi);
    findRow(epi, exceptepi, answer);
    for(int i = 0; i < answer.size(); i++) {
        for(int j = 0; j < bitnum; j++) {
            if(answer[i][j] == '2') answer[i][j] = '-';
        }
    }
    findCol(epi, exceptepi, answer, minterm);
    // 출력
    for(int i = 0; i < answer.size(); i++) {
        cout << answer[i] << " ";
    } cout << endl;
}
```

- init

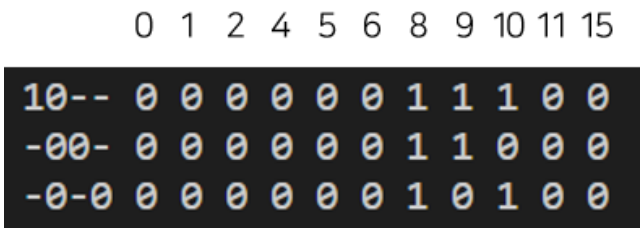
    - 입력받은 minterm을 2진수 배열로 바꾸는 함수입니다.
- deduplicateStr & deduplicateInt
    - 배열을 정렬하고 중복을 제거하기 위한 함수입니다.
- findPI
    - PI를 찾기 위한 함수입니다.
- findEPI
    - EPI를 찾기 위한 함수입니다.



EPI를 찾기 위한 matrix epi를 생성합니다. 아래의 코드는 EPI를 찾은 후 찾은 EPI와 관련된 값들에 대해 삭제하고 NEPI에 대한 표를 생성하는 과정입니다.

```
// epi 삭제 -> epi 표에서 epi 및 관련 minterm 제거
    // epi와 관련된 경우 값을 0으로
    for (int i = 0; i < epiIdx.size(); i++) {
        for (int j = 0; j < bitlist.size(); j++) {
            if (epi[epiIdx[i]][j] == 1) {
                for (int k = 0; k < epi.size(); k++) epi[k][j] = 0;
            }
        }
    }
    // epi 배열에서 전체 0인 pi 제거
    int i = 0;
    while(i != epi.size()) {
        int count = 0;
        for (int j = 0; j < epi[i].size(); j++) if (epi[i][j] == 1) count++;
        if (count == 0) {
            exceptepi.erase(exceptepi.begin()+i);
            epi.erase(epi.begin()+i);
            i = 0;
            continue;
        } i++;
    }
```

다음 결과로 epi가 NEPI로 이루어진 표로 수정되었습니다.



Row dominance

- findEPI에서 수정된 matrix epi를 이용하여 row dominance를 찾습니다.
- 위쪽부터 차례대로 하나의 NEPI와 다음 NEPI 사이에 1의 개수가 얼마나 차이나는지 확인합니다.
- 만약 0번째 NEPI가 포함하는 minterm과 다음 1번째 NEPI의 minterm이 모두 일치하고 1번째 NEPI는 추가적으로 minterm을 가진다면 row dominance 관계를 가집니다.
- 그러나 0번째 NEPI가 포함하는 minterm에 1번째 NEPI의 minterm이 모두 일치하지 않는데 1번째 NEPI가 추가적으로 minterm을 가진다면 row dominance 관계에 해당하지 않습니다.
- 모든 관계 쌍을 추적하여 중복되는 값이 있다면 먼저 선택한 row dominance 관계를 택해 출력합니다.

```
void findRow(vector<vector<int>> epi, vector<string> exceptepi, vector<string>& answer) {
    vector<pair<int, int>> rowpair;
    vector<int> row;
    vector<int> rowdelete;
    int in, out, icount, jcount;
    answer.push_back("ROW");
    if(epi.empty()) return;
    for(int i = 0; i < epi.size()-1; i++) {
        for(int j = i+1; j < epi.size(); j++) {
            in = out = icount = jcount = 0;
            for(int k = 0; k < epi[i].size(); k++) {
                if(epi[i][k] == 1) icount++;
                if(epi[j][k] == 1) jcount++;
                if(epi[i][k] == 1 && epi[j][k] == 0) in++;
                if(epi[i][k] == 0 && epi[j][k] == 1) out++;
            }
            if(icount == 0 || jcount == 0) continue;
            if(in >= 0 && out == 0) rowpair.push_back(make_pair(i, j));
            if(out > 0 && in == 0) rowpair.push_back(make_pair(j, i));
        }
    }
    if (rowpair.size() == 0) return;
    row.push_back(rowpair[0].first);
    rowdelete.push_back(rowpair[0].second);
    for(int i = 1; i < rowpair.size(); i++) {
        bool check = false;
        for(int j = 0; j < row.size(); j++) {
            if(rowpair[i].second == rowdelete[j]) check = true;
        }
        if(!check) {
            row.push_back(rowpair[i].first);
            rowdelete.push_back(rowpair[i].second);
        }
    }
    deduplicateInt(row);
    for (int i = 0; i < row.size(); i++) {
        answer.push_back(exceptepi[row[i]]);
    }
}
```

Column dominance

- findEPI에서 수정된 matrix epi를 이용하여 column dominance를 찾습니다.
- 왼쪽부터 차례대로 하나의 minterm과 다음 minterm 사이에 1의 개수가 얼마나 차이나는지 확인합니다.
- 만약 0번째 minterm을 포함하는 NEPI와 다음 1번째 minterm의 NEPI가 모두 일치하고 1번째 minterm은 추가적으로 NEPI를 가진다면 column dominance 관계를 가집니다.

- 그러나 0번째 minterm을 포함하는 NEPI에 1번째 minterm의 NEPI가 모두 일치하지 않는데 1번째 minterm을 추가적으로 포함하는 NEPI가 있다면 column dominance 관계에 해당하지 않습니다.
- 모든 관계 쌍을 추적하여 중복되는 값이 있다면 먼저 선택한 column dominance 관계를 택해 출력합니다.

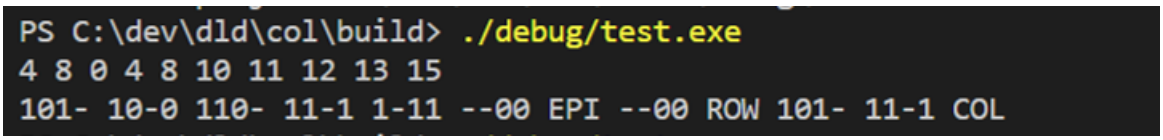
```
void findCol(vector<vector<int>> epi, vector<string> exceptepi, vector<string>& answer, vector<int> minterm) {
    vector<pair<int, int>> colpair;
    vector<int> col;
    vector<int> coldelete;
    int in, out, icount, jcount;
    answer.push_back("COL");
    if(epi.empty()) {return;}

    for(int i = 0; i < epi[0].size()-1; i++) {
        for(int j = i+1; j < epi[0].size(); j++) {
            in = out = icount = jcount = 0;
            for(int k = 0; k < epi.size(); k++) {
                if(epi[k][i] == 1) icount++;
                if(epi[k][j] == 1) jcount++;
                if(epi[k][i] == 1 && epi[k][j] == 0) in++;
                if(epi[k][i] == 0 && epi[k][j] == 1) out++;
            }
            if(icount == 0 || jcount == 0) continue;
            if(out >= 0 && in == 0) colpair.push_back(make_pair(i, j));
            if(in > 0 && out == 0) colpair.push_back(make_pair(j, i));
        }
    }
    if (colpair.size() == 0) return;
    col.push_back(colpair[0].first);
    coldelete.push_back(colpair[0].second);
    for(int i = 1; i < colpair.size(); i++) {
        bool check = false;
        for(int j = 0; j < col.size(); j++) {
            if(colpair[i].second == coldelete[j]) check = true;
        }
        if(!check) {
            col.push_back(colpair[i].first);
            coldelete.push_back(colpair[i].second);
        }
    }
    deduplicateInt(col);
    for (int i = 0; i < col.size(); i++) {
        answer.push_back(to_string(minterm[col[i]]));
    }
}
```

테스트 케이스

- CASE 1

```
// Row dominace만 해당
4 8 0 4 8 10 11 12 13 15
```



다음 테스트케이스에 해당하는 minterm에 대해 PI표를 구하면 왼쪽 그림과 같다. 오른쪽 그림은 여기서 EPI를 구한 후 NEPI에 대해 표를 작성한 것이다.

- Row dominance : 오른쪽 표에 대해 위쪽부터 순서대로 row dominance 관계를 찾아보면 101-는 10-0을 지배할 수 있고 11-1는 110-를 지배하고 있음을 알 수 있다. 따라서 출력 결과를 보면 101-와 11-1 모두 잘 찾아낸 것을 확인할 수 있다.

- CASE 2

```
// Column dominace만 해당
4 10 0 2 3 4 5 8 10 12 14 15
```

```
PS C:\dev\dld\col\build> ./debug/test.exe
4 10 0 2 3 4 5 8 10 12 14 15
001- 010- 111- 1--0 -0-0 --00 EPI 001- 010- 111- ROW COL 0
```

다음 테스트케이스에 해당하는 minterm에 대해 PI표를 구하면 왼쪽 그림과 같다. 오른쪽 그림은 여기서 EPI를 구한 후 NEPI에 대해 표를 작성한 것이다.

- Column dominance : 오른쪽 표에 대해 완쪽부터 순서대로 column dominance 관계를 찾아보면 0은 8을 지배할 수 있음을 알 수 있다. 나머지 관계에서는 column dominance 관계를 찾아볼 수 없다. 따라서 출력 결과가 0임을 확인할 수 있다.

- CASE 3

```
4 11 0 2 5 6 7 8 10 12 13 14 15
```

```
PS C:\dev\dld\col\build> ./debug/test.exe
4 11 0 2 5 6 7 8 10 12 13 14 15
11-- 1--0 -0-0 -11- -1-1 --10 EPI -0-0 -1-1 ROW 11-- -11- COL 6
```

다음 테스트케이스에 해당하는 minterm에 대해 PI표를 구하면 왼쪽 그림과 같다. 오른쪽 그림은 여기서 EPI를 구한 후 NEPI에 대해 표를 작성한 것이다.

- Row dominance : 오른쪽 표에 대해 위쪽부터 순서대로 row dominance 관계를 찾아보면 11—와 1—0은 interchangeable 관계에 있음을 알 수 있다. 위쪽에 있는 11—를 row dominance로 결정하고 계속해서 다음 관계를 찾는다. 이때 -11-과 —10 또한 interchangeable 관계로, 앞에 있는 -11-를 row dominance로 결정했다. 결과적으로 11—와 -11-가 row dominance로 잘 출력되고 있음을 확인할 수 있다.
- Column dominance : 오른쪽 표에 대해 완쪽부터 순서대로 column dominance 관계를 찾아보면 6은 14를 지배할 수 있다. 12 또한 14를 지배할 수 있지만 6을 먼저 column dominance로 택했기 때문데 출력이 6이 되는 것을 알 수 있다.

- CASE 4

```
4 11 0 1 2 4 5 6 8 9 10 11 15
```

```
PS C:\dev\dld\col\build> ./debug/test.exe
4 11 0 1 2 4 5 6 8 9 10 11 15
0-0- 0--0 10-- 1-11 -00- -0-0 EPI 0-0- 0--0 1-11 ROW 10-- COL 9
```

다음 테스트케이스에 해당하는 minterm에 대해 PI표를 구하면 왼쪽 그림과 같다. 오른쪽 그림은 여기서 EPI를 구한 후 NEPI에 대해 표를 작성한 것이다.

- Row dominance : 오른쪽 표에 대해 위쪽부터 순서대로 row dominance 관계를 찾아보면 10—는 -00-와 -0-0을 모두 지배하고 있음을 알 수 있다.
- Column dominance : 오른쪽 표에 대해 완쪽부터 순서대로 column dominance 관계를 찾아보면 9가 8을 지배하고 있음을 알 수 있다. 10 또한 8을 지배할 수 있지만 앞서 9가 먼저 선택됐기 때문에 결과적으로 column dominance는 9 하나가 된다.