

1. Intro. This is a simple filter that inputs the well-known DIMACS format for satisfiability problems and outputs the format used by SAT0, SAT1, etc.

DIMACS format begins with zero or more optional comment lines, indicated by their first character ‘c’. The next line should say ‘p cnf n m ’, where n is the number of variables and m is the number of clauses. Then comes a string of m “clauses,” which are sequences of nonzero integers of absolute value $\leq n$, followed by zero. A literal for the k th variable is represented by k ; its complement is represented by $-k$.

SAT format is explained in the programs cited above.

```
#define buf_size 1024
#include <stdio.h>
#include <stdlib.h>
char buf[buf_size];
int m, n, v;
main()
{
    register int j, k, l, p;
    <Read the preamble 2>;
    if (m == 0 || n == 0) {
        fprintf(stderr, "I didn't find m or n!\n");
        exit(-1);
    }
    <Read and translate the clauses 4>;
}
```

2. <Read the preamble 2> \equiv

```
while (1) {
    if (!fgets(buf, buf_size, stdin)) break;
    if (buf[0] == 'c') <Process a comment line and continue 3>;
    if (buf[0] != 'p' || buf[1] != ' ' || buf[2] != 'c' || buf[3] != 'n' || buf[4] != ' ') {
        fprintf(stderr, "Unrecognized input line: %s\n", buf);
        exit(-2);
    }
    sscanf(buf + 5, "%i%i", &n, &m);
    break;
}
```

This code is used in section 1.

3. <Process a comment line and continue 3> \equiv

```
{
    printf("~%s", buf + 1);
    continue;
}
```

This code is used in section 2.

```

4.  ⟨ Read and translate the clauses 4 ⟩ ≡
    j = k = l = p = 0;
    while (fscanf(stdin, "%i", &v) == 1) {
        if (v == 0) {
            if (j == 0) fprintf(stderr, "Warning: Clause %d is empty!\n", k + 1);
            printf("\n");
            if (k == m) {
                fprintf(stderr, "Too many clauses (more than %d)!\n", m);
                exit(-3);
            }
            k++, j = 0;
        } else if (v > 0) {
            if (v > n) {
                fprintf(stderr, "Too many variables (%d > %d)!\n", v, n);
                exit(-4);
            }
            printf("_%d", v);
            if (v > p) p = v;
            j++, l++;
        } else {
            if (v < -n) {
                fprintf(stderr, "Too many variables (%d < -%d)!\n", v, n);
                exit(-5);
            }
            printf("_~%d", -v);
            if (-v > p) p = -v;
            j++, l++;
        }
    }
    if (j) {
        fprintf(stderr, "The last clause didn't end with 0!\n");
        printf("\n");
        k++;
    }
    if (k < m) fprintf(stderr, "Too few clauses (%d < %d)!\n", k, m);
    fprintf(stderr, "OK, I wrote out %d literals in %d clauses on %d variables.\n", l, k, p);

```

This code is used in section 1.

5. Index.

buf: [1](#), [2](#), [3](#).

buf_size: [1](#), [2](#).

exit: [1](#), [2](#), [4](#).

fgets: [2](#).

fprintf: [1](#), [2](#), [4](#).

fscanf: [4](#).

j: [1](#).

k: [1](#).

l: [1](#).

m: [1](#).

main: [1](#).

n: [1](#).

p: [1](#).

printf: [3](#), [4](#).

sscanf: [2](#).

stderr: [1](#), [2](#), [4](#).

stdin: [2](#), [4](#).

v: [1](#).

⟨ Process a comment line and **continue** 3 ⟩ Used in section 2.
⟨ Read and translate the clauses 4 ⟩ Used in section 1.
⟨ Read the preamble 2 ⟩ Used in section 1.

DIMACS-TO-SAT

	Section	Page
Intro	1	1
Index	5	3