# Overview

## Implementation details

The program reads the input file line by line and parses it with space as a delimiter. All the variables are mapped to distinct integers. The sentence is then encoded into bits (in a long int variable); the bit values represent the presence of a variable in the sentence and its sign (if negated). Each input and derived sentence is associated with an identification number. When two sentences can be resolved using the resolution operator, they are pushed into a priority queue (as explained in the problem description). Sentences in the queue are resolved in the order, for every newly generated sentence, it is compared with all the existing sentences to determine if it can be resolved with some sentence, if so, the pair is pushed in the queue.

## Important data-structures used

1. indexMap: This stores the mapping of a variable/symbol to its identification number.
2. symbolMap: This is the reverse mapping of indexMap.
3. kb: This stores the mapping of encoded sentence to its identification number.
4. kb2: This is the reverse mapping of kb.
5. parents: This stores the mapping of a derived sentence id to the sentences from which it was derived.

## Limitations

1. Since the queue length could get really large, I limit the queue size to 10^6. If no empty statement is obtained before this length is reached, the program returns ‘entailment failure’.
2. The implementation works for up to 32 variables in the input.

# Program Outputs

## Sammy’s sport shop (for input given in the problem)

success!

29. -C3B -L3B [input]

35. L3B [input]

39. -C3B [29, 35]

32. O3Y [input]

19. C3Y C3B -O3Y [input]

7. -C1Y -C3Y [input]

22. -C1W -L1W [input]

33. L1W [input]

37. -C1W [22, 33]

3. C1Y C1W C1B [input]

10. -C1B -C2B [input]

24. -C2Y -L2Y [input]

34. L2Y [input]

38. -C2Y [24, 34]

4. C2Y C2W C2B [input]

36. -C2W [input]

40. C2Y C2B [4, 36]

41. C2B [38, 40]

42. -C1B [10, 41]

45. C1Y C1W [3, 42]

46. C1Y [37, 45]

47. -C3Y [7, 46]

64. C3B -O3Y [19, 47]

65. C3B [32, 64]

66. [39, 65]

# iterations : 47

Max queue size : 170

## Example1.kb

success!

1. -R -A [input]

2. A [input]

6. -R [1, 2]

5. -S [input]

3. P [input]

0. -P -Q R S [input]

4. Q [input]

7. -P R S [0, 4]

8. R S [3, 7]

9. R [5, 8]

10. [6, 9]

# iterations : 5

Max queue size : 11