ANISH DUBEY(523002969)
Artificial Intelligence Assignment-4

Q) Include a document that details how to compile and run your code.
A) Readme file included.

Q) Provide a brief overview of how your program works, including significant data structures and algorithmic details (such as how you implemented the heuristic or visited list), and how they might impact the performance of your program?
A)
The program takes CNF as an input from the kb.txt file. It creates an object for each clauses and store in a vector. Also isresolvable() and resolve() functions are written for checking if 2 strings can be resolved or not. If yes, resolve() will give the resolved output.
Then all possible pairs of resolvable clauses are inserted in a priority queue based on length of 2 clauses. I have created a class for the pairs of clauses (repair). New clauses are generated from old clauses and if not visited, they are kept in vector of clauses. Also new respair object which contains pair of clauses are inserted in priority queue. This process is repeated until empty clause is not found or till 10000 iterations are reached.
I have created **classes** for
i) **Clauses class**: It contains clause id, parents from which it is extracted. In case of input parents are set to NULL.
ii) **Repair class:** It contains two clauses.

Data Structures:
i) **Priority Queue:** I have used standard c++ priority queue. Comparator function has been over ridden by me according to the length of the clauses. So when 2 clauses of same length needs to be pushed in priority queue, here randomness comes and iteration varies from 17 to 38 in my case for sammy sport problem.
ii) **Visited:** I have mapped all labels, observations and contains into alphabets character. I have used map for that. Reason for doing is to keep track of visited clauses. Like “L1W” is represented as “a” and “-L1W” as “A”. After mapping , string is sorted and kept in a map to check whether it has been visited or not.
iii) **Heuristic Implementation:** Heuristic is implemented as said in the question. Clause pairs are inserted in the priority queue based on length of the pair of clauses.

Important Functions:
i)**isresolvable()**: It checks whether two clauses will be resolvable or not. If yes, then what all literals can be resolved.
ii) **resolve():** Returns the resolved clause.

Q)Provide a text document with your knowledge base (propositional rules) for Sammy's Sport Shop.
A) Knowledge Base text file included

Q)Provide the input file for Sammy's Sport Shop, with all the rules and facts converted to CNF?

A) kb.txt is provided in the folder.

Q)**Give a transcript of your program solving Sammy's Sport Shop, showing a trace of all the resolution steps and a trace of the proof when the empty clause is found.**
Ans)

```
Iteration 1 , Queue Size 167
Resolution on -L1W -C1W(0) and L1W(57) resulted in -C1W

Iteration 2 , Queue Size 168
Resolution on -L2Y -C2Y(4) and L2Y(58) resulted in -C2Y

Iteration 3 , Queue Size 169
Resolution on -L3B -C3B(8) and L3B(59) resulted in -C3B

Iteration 4 , Queue Size 171
Resolution on -O3Y C3Y C3B(52) and O3Y(56) resulted in C3Y C3B

Iteration 5 , Queue Size 187
Resolution on -C3B(63) and C3Y C3B(64) resulted in C3Y

Iteration 6 , Queue Size 195
Resolution on -L3Y -C3Y(7) and C3Y(65) resulted in -L3Y

Iteration 7 , Queue Size 194
Resolution on -C3B -C3Y(25) and C3Y(65) resulted in -C3B

Iteration 8 , Queue Size 193
Resolution on -C3W -C3Y(27) and C3Y(65) resulted in -C3W

Iteration 9 , Queue Size 194
Resolution on -C1Y -C3Y(33) and C3Y(65) resulted in -C1Y

Iteration 10 , Queue Size 195
Resolution on -C2Y -C3Y(39) and C3Y(65) resulted in -C2Y

Iteration 11 , Queue Size 194
Resolution on -C3Y -C1Y(44) and C3Y(65) resulted in -C1Y

Iteration 12 , Queue Size 193
Resolution on -C3Y -C3W(29) and C3Y(65) resulted in -C3W

Iteration 13 , Queue Size 192
Resolution on -C3Y -C3B(28) and C3Y(65) resulted in -C3B

Iteration 14 , Queue Size 191
Resolution on -C3Y -C2Y(45) and C3Y(65) resulted in -C2Y
```

```
Iteration 15 , Queue Size 190
Resolution on -O3Y C3Y C3B(52) and -C3B(63) resulted in -O3Y C3Y

Iteration 16 , Queue Size 199
Resolution on O3Y(56) and -O3Y C3Y(69) resulted in C3Y

Iteration 17 , Queue Size 198
Resolution on -O1Y C1Y C1B(48) and O1Y(54) resulted in C1Y C1B

Iteration 18 , Queue Size 214
Resolution on -C1Y(68) and C1Y C1B(70) resulted in C1B

Iteration 19 , Queue Size 222
Resolution on -L1B -C1B(2) and C1B(71) resulted in -L1B

Iteration 20 , Queue Size 221
Resolution on -C1W -C1B(14) and C1B(71) resulted in -C1W

Iteration 21 , Queue Size 220
Resolution on -C1Y -C1B(16) and C1B(71) resulted in -C1Y

Iteration 22 , Queue Size 219
Resolution on -C1B -C2B(34) and C1B(71) resulted in -C2B

Iteration 23 , Queue Size 221
Resolution on -C1B -C1W(12) and C1B(71) resulted in -C1W

Iteration 24 , Queue Size 220
Resolution on -C1B -C1Y(13) and C1B(71) resulted in -C1Y

Iteration 25 , Queue Size 219
Resolution on -C2B -C1B(40) and C1B(71) resulted in -C2B

Iteration 26 , Queue Size 218
Resolution on -C1B -C3B(35) and C1B(71) resulted in -C3B

Iteration 27 , Queue Size 217
Resolution on -L3B -C3B(8) and C3Y C3B(64) resulted in -L3B C3Y

Iteration 28 , Queue Size 226
Resolution on L3B(59) and -L3B C3Y(74) resulted in C3Y

Iteration 29 , Queue Size 225
Resolution on -C2B -C3B(41) and C3Y C3B(64) resulted in -C2B C3Y
```

```
Iteration 30 , Queue Size 236
Resolution on -C3W -C3Y(27) and -L3B C3Y(74) resulted in -C3W -L3B

Iteration 31 , Queue Size 238
Resolution on L3B(59) and -C3W -L3B(76) resulted in -C3W

Iteration 32 , Queue Size 237
Resolution on -O1Y C1Y C1B(48) and -C1Y(68) resulted in -O1Y C1B

Iteration 33 , Queue Size 246
Resolution on O1Y(54) and -O1Y C1B(77) resulted in C1B

Iteration 34 , Queue Size 245
Resolution on -C3Y -C3B(28) and -L3B C3Y(74) resulted in -C3B -L3B

Iteration 35 , Queue Size 244
Resolution on -C3B -C3W(24) and C3Y C3B(64) resulted in -C3W C3Y

Iteration 36 , Queue Size 254
Resolution on C2B C2W C2Y(10) and -C2B(73) resulted in C2W C2Y

Iteration 37 , Queue Size 271
Resolution on -C2W(60) and C2W C2Y(79) resulted in C2Y

Iteration 38 , Queue Size 280
Resolution on -C2Y(62) and C2Y(80) resulted in
```

Printing Trace

```
81: [] [80 ,62 ]
 80: C2Y [79 ,60 ]
  79: C2W C2Y [73 ,10 ]
   73: -C2B [71 ,34 ]
    71: C1B [70 ,68 ]
     70: C1Y C1B [54 ,48 ]
      54: O1Y [input]
      48: -O1Y C1Y C1B [input]
     68: -C1Y [65 ,33 ]
      65: C3Y [64 ,63 ]
       64: C3Y C3B [56 ,52 ]
        56: O3Y [input]
        52: -O3Y C3Y C3B [input]
       63: -C3B [59 ,8 ]
        59: L3B [input]
        8: -L3B -C3B [input]
      33: -C1Y -C3Y [input]
    34: -C1B -C2B [input]
   10: C2B C2W C2Y [input]
  60: -C2W [input]
 62: -C2Y [58 ,4 ]
  58: L2Y [input]
  4: -L2Y -C2Y [input]
Resolved
```

Q)Report the total number of resolutions (iterations of the main loop) and max queue size.
A)Iterations: Iterations vary from 17 to 38 depending on the randomness obtained in priority queue when equal size clause length are entered in the priority queue.
Queue Size:280

Q)(You might also want to optionally include a trace of another inference example, such as example1.kb above, or perhaps 'safe22' in the Wumpus World.)
A) Printing trace of example1.kb

```
Iteration 1 , Queue Size 5
Resolution on -A -R(1) and A(2) resulted in -R

Iteration 2 , Queue Size 5
Resolution on -P -Q R S(0) and Q(4) resulted in -P R S

Iteration 3 , Queue Size 8
Resolution on P(3) and -P R S(7) resulted in R S

Iteration 4 , Queue Size 10
Resolution on -S(5) and R S(8) resulted in R

Iteration 5 , Queue Size 11
Resolution on -R(6) and R(9) resulted in

Printing Trace
10:  [] [9 ,6 ]
 9: R [8 ,5 ]
  8: R S [7 ,3 ]
   7: -P R S [4 ,0 ]
    4: Q [input]
    0: -P -Q R S [input]
   3: P [input]
  5: -S [input]
 6: -R [2 ,1 ]
  2: A [input]
  1: -A -R [input]
Resolved
```