Cardiovascular Disease Diagnosis Expert System

Implemented using Backward and Forward chaining principles

Kody Gentry, Levi, Tim George

AI FALL 2021

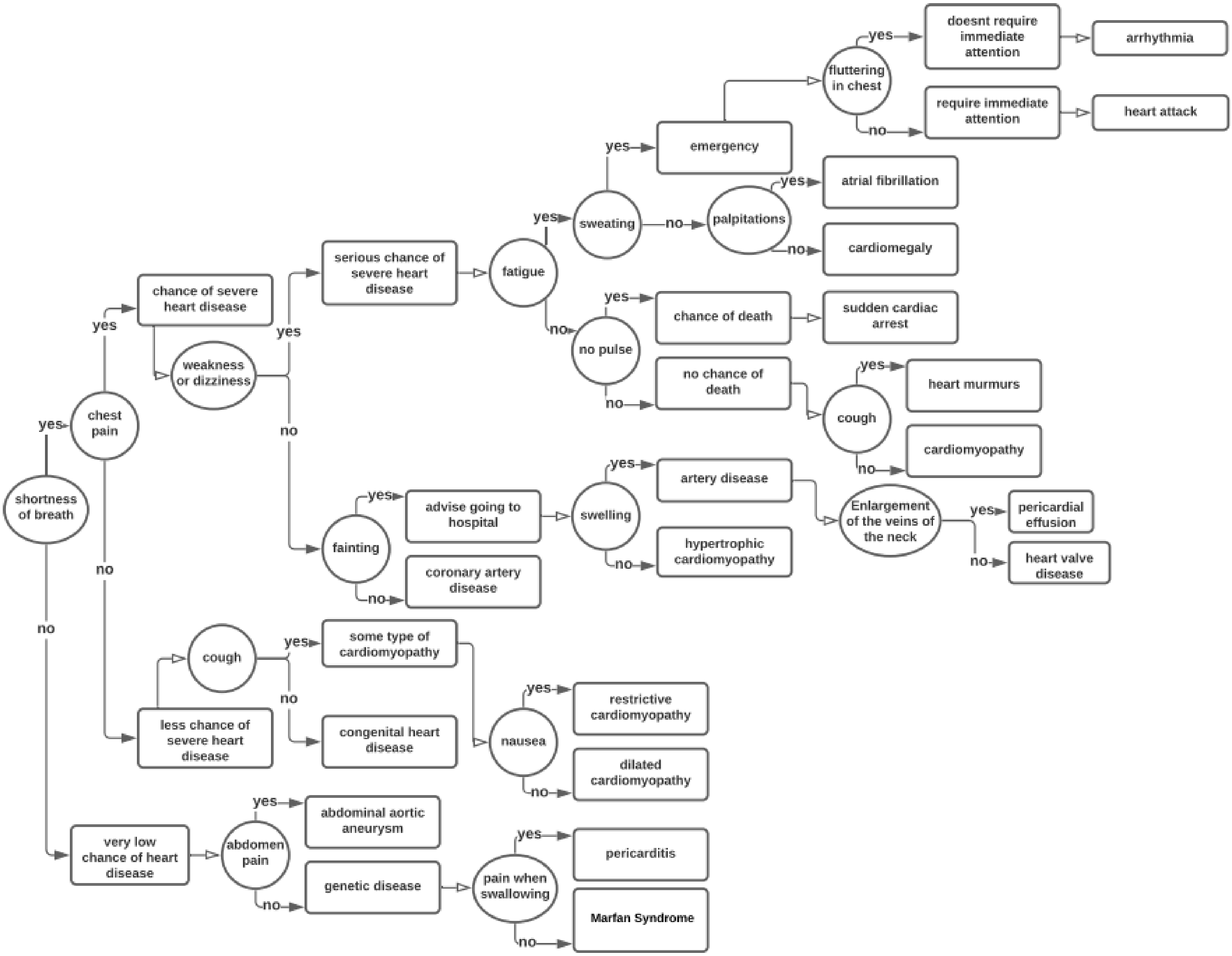
INDEX

Our problem is to supply hospitals a way to automate the process of the diagnosis of Cardiovascular (Heart) diseases quickly and efficiently and to recommend the treatment based on the diagnosis. Not every patient can be seen promptly by a doctor in person, so we created a way for patients to get the answers they need with just a few clicks. Our solution is an intelligent computer expert system. An expert system is a computer program that uses artificial intelligence technologies to simulate the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field, which, in our case would be a doctor. We must implement the expert system program by employing two concepts: Backward Chaining for diagnosis, and Forward Chaining for determining treatments. Our domain revolves strictly around *heart* diseases and their related symptoms and treatments, but our system does not reach anything beyond that scope.

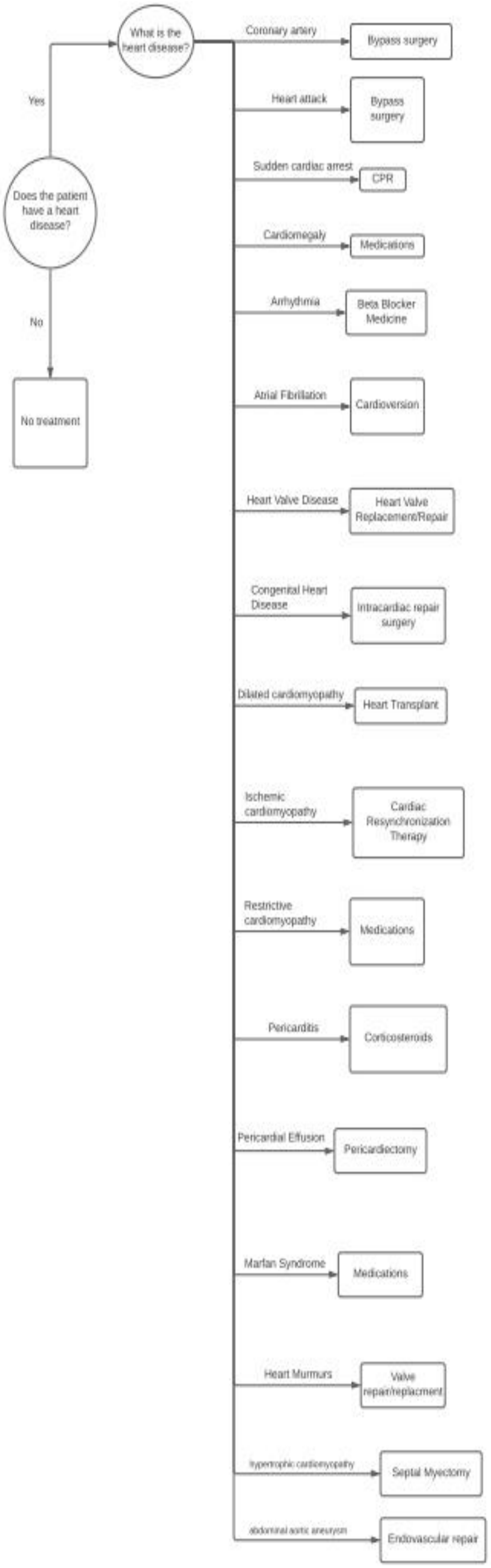
Backward chaining is a concept in artificial intelligence that involves backtracking from the endpoint or goal to steps/nodes that led to the endpoint. This type of chaining starts from the goal and moves backward to comprehend the steps that were taken to attain this goal. The backtracking process can also enable a person to establish logical steps that can be used to find other important solutions. Simply put, it's a goal-driven method of reasoning that we employed through an inference engine. The basis of backward chaining is referred to as *Modus Ponens*, meaning if both the conditional statement (x->y) and the antecedent (x) are true, then we can infer the subsequent (y).

Forward chaining is a method of reasoning in artificial intelligence in which inference rules are applied to existing data to output additional data until an endpoint/goal is achieved. In this type of chaining, the inference engine starts by evaluating given facts and conditions before deriving new information. An endpoint is achieved through the manipulation of knowledge that exists in the knowledge base.

BACKWARD CHAINING DECISION TREE



FORWARD CHAINING DECISION TREE



BACKWARD CHAINING RULES

```
10 IF SHORTNESS_OF_BREATH = NO
        THEN LOW_CHANCE_OF_HEART_DISEASE = YES
20 IF LOW_CHANCE_OF_HEART_DISEASE = YES AND ABDOMEN_PAIN = YES
        THEN DISEASE = abdominal aortic aneurysm
30 IF LOW_CHANCE_OF_HEART_DISEASE = YES AND ABDOMEN_PAIN = NO
        THEN GENETIC_DISEASE = YES
40 IF GENETIC_DISEASE = YES AND PAIN_SWALLOWING = YES
        THEN DISEASE = pericarditis
50 IF GENETIC_DISEASE = YES AND PAIN_SWALLOWING = NO
        THEN DISEASE = Marfan Syndrome
60 IF SHORTNESS_OF_BREATH = YES AND CHEST_PAIN = NO
        THEN LESS_CHANCE_OF_SEVERE_HEART_DISEASE = YES
70 IF LESS_CHANCE_OF_SEVERE_HEART_DISEASE = YES AND COUGH = NO
        THEN DISEASE = congenital heart disease
80 IF LESS_CHANCE_OF_SEVERE_HEART_DISEASE = YES AND COUGH = YES
        THEN TYPE_OF_CARDIOMYOPATHY = YES
90 IF TYPE_OF_CARDIOMYOPATHY = YES AND NAUSEA = NO
        THEN DISEASE = dilated cardiomyopathy
100 IF TYPE_OF_CARDIOMYOPATHY = YES AND NAUSEA = YES
        THEN DISEASE = restrictive cardiomyopathy
110 IF SHORTNESS_OF_BREATH = YES AND CHEST_PAIN = YES
        THEN CHANCE_OF_SEVERE_HEART_DISEASE = YES
```

120 IF CHANCE_OF_SEVERE_HEART_DISEASE = YES AND WEAKNESS_DIZZINESS = NO AND FAINTING = NO

THEN DISEASE = coronary artery disease

130 IF CHANCE_OF_SEVERE_HEART_DISEASE = YES AND WEAKNESS_DIZZINESS = NO AND FAINTING = YES

THEN ADVISE_HOSPITAL = YES

140 IF ADVISE_HOSPITAL = YES AND SWELLING = NO

THEN DISEASE = hypertrophic cardiomyopathy

150 IF ADVISE_HOSPITAL = YES AND SWELLING = YES

THEN ARTERY_DISEASE = YES

160 IF ARTERY_DISEASE = YES AND VEIN_ENLARGMENT_NECK = NO

THEN DISEASE = heart valve disease

170 IF ARTERY_DISEASE = YES AND VEIN_ENLARGMENT_NECK = YES

THEN DISEASE = pericardial effusion

180 IF CHANCE_OF_SEVERE_HEART_DISEASE = YES AND WEAKNESS_DIZZINESS = YES

THEN SERIOUS_CHANCE_OF_SEVERE_HEART_DISEASE = YES

190 IF SERIOUS_CHANCE_OF_SEVERE_HEART_DISEASE = YES AND FATIGUE = NO AND NO_PULSE = YES

THEN CHANCE_OF_DEATH = YES

200 IF CHANCE_OF_DEATH = YES

THEN DISEASE = sudden cardiac arrest

210 IF SERIOUS_CHANCE_OF_SEVERE_HEART_DISEASE = YES AND FATIGUE = NO AND NO_PULSE = NO

THEN NO_CHANCE_OF_DEATH = YES

220 IF NO_CHANCE_OF_DEATH = YES AND COUGH = YES

THEN DISEASE = heart murmurs

230 IF NO_CHANCE_OF_DEATH = YES AND COUGH = NO

THEN DISEASE = cardiomyopathy

240 IF SERIOUS_CHANCE_OF_SEVERE_HEART_DISEASE = YES AND FATIGUE = YES AND SWEATING = NO AND PALPITATIONS = YES

THEN DISEASE = Atrial Fibrillation

250 IF SERIOUS_CHANCE_OF_SEVERE_HEART_DISEASE = YES AND FATIGUE = YES AND SWEATING = NO AND PALPITATIONS = NO

THEN DISEASE = cardiomegaly

260 IF SERIOUS_CHANCE_OF_SEVERE_HEART_DISEASE = YES AND FATIGUE = YES AND SWEATING = YES

THEN EMERGENCY = YES

270 IF EMERGENCY = YES AND FLUTTERING_IN_CHEST = NO

THEN REQUIRE_IMMEDIATE_ATTENTION = YES

280 IF REQUIRE_IMMEDIATE_ATTENTION = YES

THEN DISEASE = heart attack

290 IF EMERGENCY = YES AND FLUTTERING_IN_CHEST = YES

THEN DONT_REQUIRE_IMMEDIATE_ATTENTION = YES

300 IF DONT_REQUIRE_IMMEDIATE_ATTENTION = YES

THEN DISEASE = arrhythmia

FORWARD CHAINING RULES

10 IF HEARTDISEASE = NO

THEN TREATMENT = NONE

20 IF HEARTDISEASE = YES AND DISEASENAME = Coronary Artery

THEN TREATMENT = Bypass Surgery

30 IF HEARTDISEASE = YES AND DISEASENAME = Heart Attack

THEN TREATMENT = Bypass Surgery

40 IF HEARTDISEASE = YES AND DISEASENAME = Sudden Cardiac Arrest

THEN TREATMENT = CPR

50 IF HEARTDISEASE = YES AND DISEASENAME = Cardiomegaly

THEN TREATMENT = Medications

60 IF HEARTDISEASE = YES AND DISEASENAME = Arrhythmia

THEN TREATMENT = Beta Blocker Medicine

70 IF HEARTDISEASE = YES AND DISEASENAME = Atrial Fibrillation

THEN TREATMENT = Cardioversion

80 IF HEARTDISEASE = YES AND DISEASENAME = heart valve disease

THEN TREATMENT = Heart Valve Replacement/Repair

90 IF HEARTDISEASE = YES AND DISEASENAME = congenital heart disease

THEN TREATMENT = Intracardiac repair surgery

```
100 IF HEARTDISEASE = YES AND DISEASENAME = Dilated cardiomyopathy
    THEN TREATMENT = Heart Transplant
110 IF HEARTDISEASE = YES AND DISEASENAME = Ischemic cardiomyopathy
    THEN TREATMENT = Cardiac Resynchronization Therapy
120 IF HEARTDISEASE = YES AND DISEASENAME = Restrictive cardiomyopathy
    THEN TREATMENT = Medications
130 IF HEARTDISEASE = YES AND DISEASENAME = Pericarditis
    THEN TREATMENT = Corticosteroids
140 IF HEARTDISEASE = YES AND DISEASENAME = Pericardial Effusion
    THEN TREATMENT = Pericardiectomy
150 IF HEARTDISEASE = YES AND DISEASENAME = Marfan Syndrome
    THEN TREATMENT = Medications
160 IF HEARTDISEASE = YES AND DISEASENAME = Heart Murmurs
    THEN TREATMENT = Valve repair/replacment
170 IF HEARTDISEASE = YES AND DISEASENAME = hypertrophic cardiomyopathy
    THEN TREATMENT = Septal Myectomy
180 IF HEARTDISEASE = YES AND DISEASENAME = abdominal aortic aneurysm
    THEN TREATMENT = Endovascular repair
190 IF HEARTDISEASE = YES AND DISEASENAME = cardiomyopathy
    THEN TREATMENT = Surgery
```

IMPLEMENTATION

Backward chaining:

First we instantiate the clause variable list, the variable map and conclusion table. We then search the conclusion table for a disease, then push variables onto the conclusion stack with rule number associated with it. We loop through the following while no heart disease is found or if we hit the end of the conclusion table. If a disease is found, push disease rule to stack, search conclusion variable list. Break if every variable is instantiated or if variable map and variable list match. If sub conclusion matches push rule to stack and continue, then search variable list for that conclusion and instantiate variable where it left off. The program then recurses all the way back to root note, to make sure all rules are instantiated. Then it checks the rule list and pops off all conclusions off stack and checks the rules to see if they are instantiated. Then determine if heart disease exists, if the stack is empty and no disease is found then no disease, otherwise return disease found.

Forward chaining:

First we define the clause variable list, variable list, and instantiated list. Then we find the condition: “has heart disease(y/n)”. This condition variable is placed on the queue. Then fetch the disease name from backward chaining, then search clause variable list for said variable. If variable found, place rule number and clause number associated with the disease and set to variable pointer. Then match found disease to treatment and add to clause conclusion variable queue and return first position in queue.

Main:

Our driver file links these two functions together. First we use <chrono> to start the clock for the execution time. Then we initialize the variable list, clause variable list, and conclusion list. We then call the backward chaining inference engine to give us a disease and use that to provide input to the forward chaining function, which then returns treatment and ending the execution clock, displaying the disease diagnosis, the recommended treatment and total execution time.

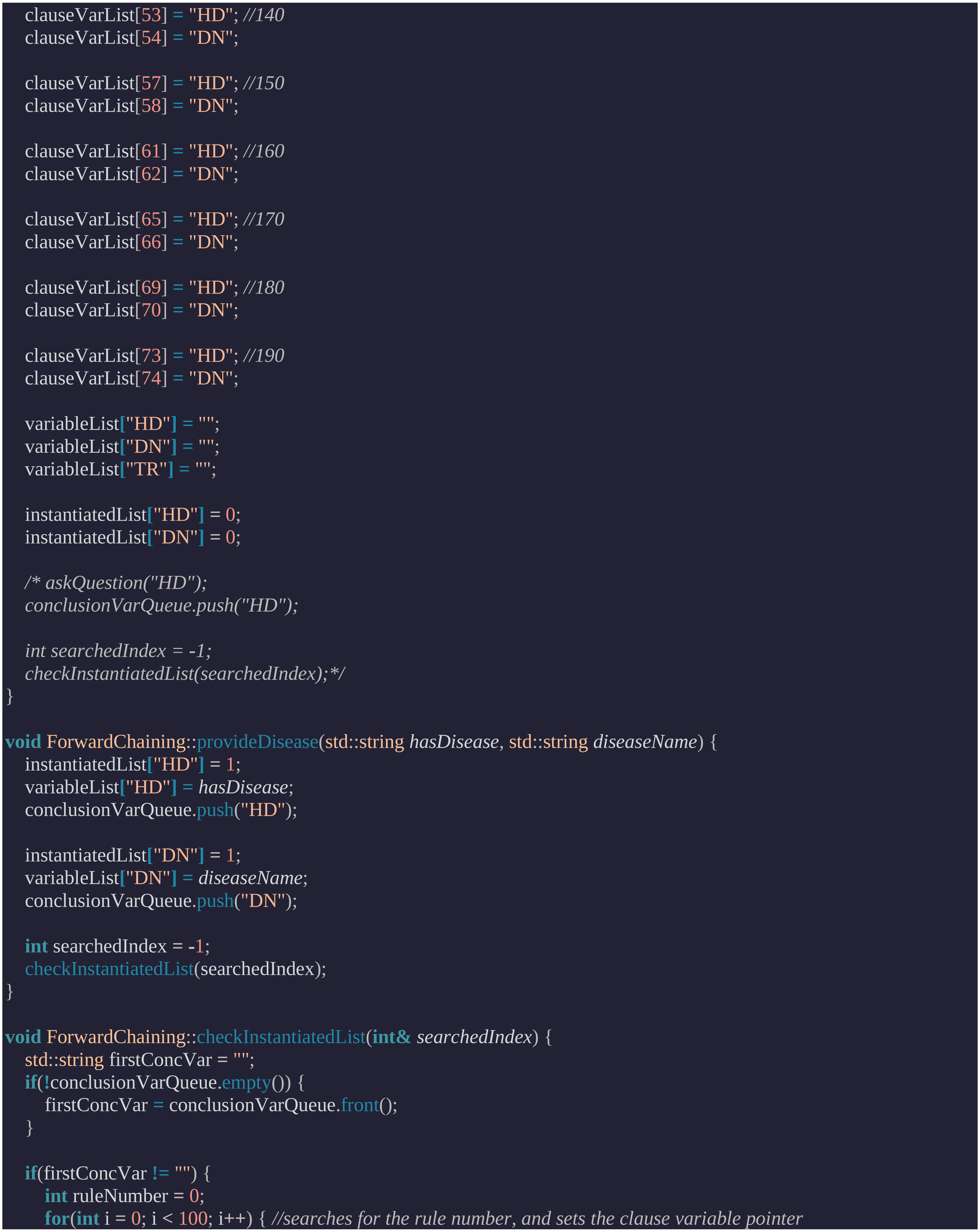
SOURCE CODE

Forward chaining class declaration file

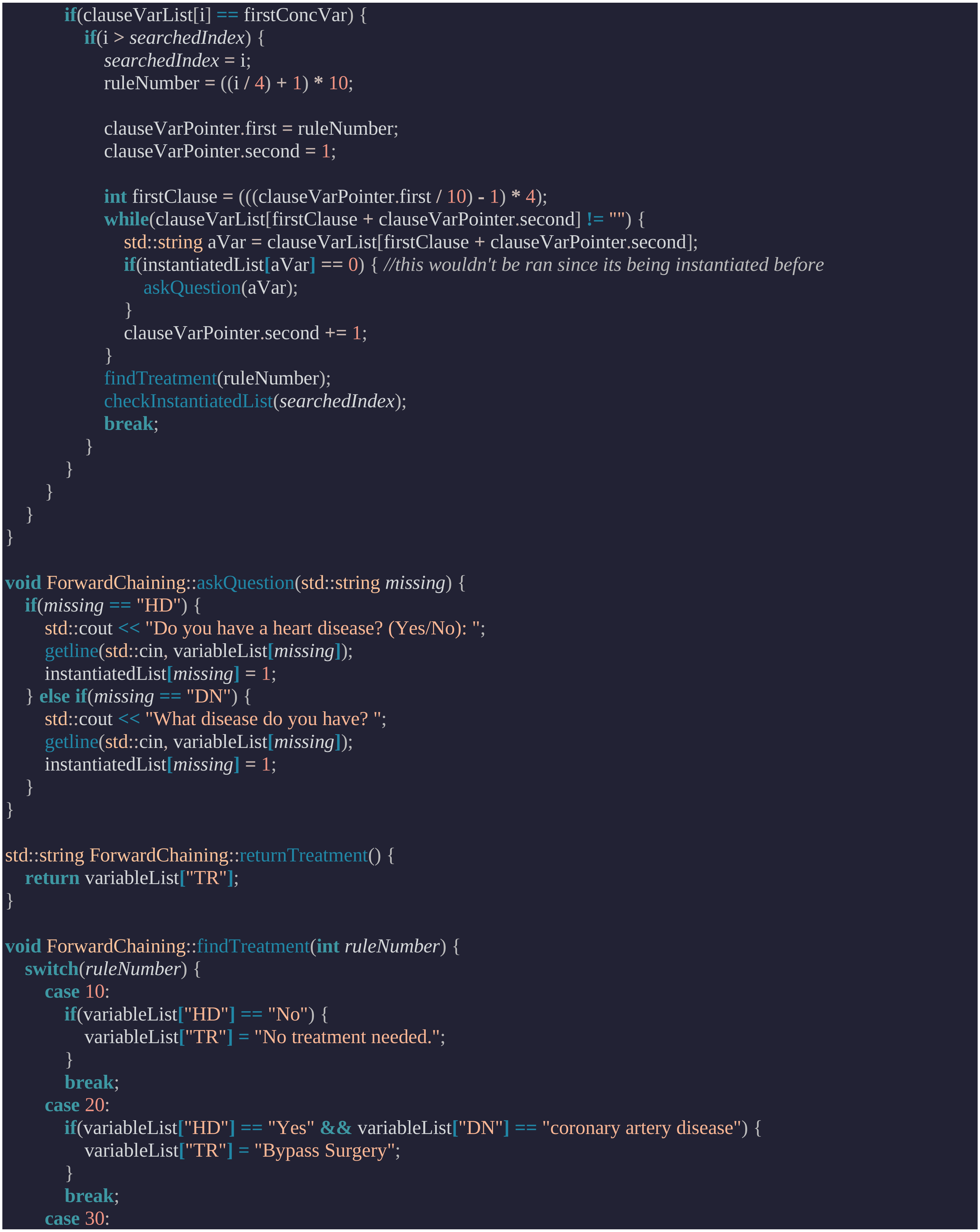
```
#ifndef FORWARD_CHAINING_H
#define FORWARD_CHAINING_H
#include <iostream>
#include <string>
#include <queue>
#include <utility>
#include <unordered_map>

class ForwardChaining
{
private:
    std::string clauseVarList[200];
    std::queue<std::string> conclusionVarQueue;
    std::pair<int, int> clauseVarPointer; //rule number, clause number
    std::unordered_map<std::string, int> instantiatedList;
    std::unordered_map<std::string, std::string> variableList;

public:
    ForwardChaining();
    void init();
    void provideDisease(std::string, std::string);
    void checkInstantiatedList(int&);
    void askQuestion(std::string);
    std::string returnTreatment();
    void findTreatment(int);

};

#endif
•
```

Forward chaining class definition file

```
#include <string>
#include <iostream>
#include "forward_chaining.h"

ForwardChaining::ForwardChaining(){

}

void ForwardChaining::init() {
    for(int i = 0; i < 100; i++) {
        clauseVarList[i] = "";
    }

    clauseVarList[1] = "HD"; //10

    clauseVarList[5] = "HD"; //20
    clauseVarList[6] = "DN";

    clauseVarList[9] = "HD"; //30
    clauseVarList[10] = "DN";

    clauseVarList[13] = "HD"; //40
    clauseVarList[14] = "DN";

    clauseVarList[17] = "HD"; //50
    clauseVarList[18] = "DN";

    clauseVarList[21] = "HD"; //60
    clauseVarList[22] = "DN";

    clauseVarList[25] = "HD"; //70
    clauseVarList[26] = "DN";

    clauseVarList[29] = "HD"; //80
    clauseVarList[30] = "DN";

    clauseVarList[33] = "HD"; //90
    clauseVarList[34] = "DN";

    clauseVarList[37] = "HD"; //100
    clauseVarList[38] = "DN";

    clauseVarList[41] = "HD"; //110
    clauseVarList[42] = "DN";

    clauseVarList[45] = "HD"; //120
    clauseVarList[46] = "DN";

    clauseVarList[49] = "HD"; //130
    clauseVarList[50] = "DN";
```



```
    clauseVarList[53] = "HD"; //140
    clauseVarList[54] = "DN";

    clauseVarList[57] = "HD"; //150
    clauseVarList[58] = "DN";

    clauseVarList[61] = "HD"; //160
    clauseVarList[62] = "DN";

    clauseVarList[65] = "HD"; //170
    clauseVarList[66] = "DN";

    clauseVarList[69] = "HD"; //180
    clauseVarList[70] = "DN";

    clauseVarList[73] = "HD"; //190
    clauseVarList[74] = "DN";

    variableList["HD"] = "";
    variableList["DN"] = "";
    variableList["TR"] = "";

    instantiatedList["HD"] = 0;
    instantiatedList["DN"] = 0;

    /* askQuestion("HD");
    conclusionVarQueue.push("HD");

    int searchedIndex = -1;
    checkInstantiatedList(searchedIndex);*/
}

void ForwardChaining::provideDisease(std::string hasDisease, std::string diseaseName) {
    instantiatedList["HD"] = 1;
    variableList["HD"] = hasDisease;
    conclusionVarQueue.push("HD");

    instantiatedList["DN"] = 1;
    variableList["DN"] = diseaseName;
    conclusionVarQueue.push("DN");

    int searchedIndex = -1;
    checkInstantiatedList(searchedIndex);
}

void ForwardChaining::checkInstantiatedList(int& searchedIndex) {
    std::string firstConcVar = "";
    if(!conclusionVarQueue.empty()) {
        firstConcVar = conclusionVarQueue.front();
    }

    if(firstConcVar != "") {
        int ruleNumber = 0;
        for(int i = 0; i < 100; i++) { //searches for the rule number, and sets the clause variable pointer
```



```
            if(clauseVarList[i] == firstConcVar) {
                if(i > searchedIndex) {
                    searchedIndex = i;
                    ruleNumber = ((i / 4) + 1) * 10;

                    clauseVarPointer.first = ruleNumber;
                    clauseVarPointer.second = 1;

                    int firstClause = (((clauseVarPointer.first / 10) - 1) * 4);
                    while(clauseVarList[firstClause + clauseVarPointer.second] != "") {
                        std::string aVar = clauseVarList[firstClause + clauseVarPointer.second];
                        if(instantiatedList[aVar] == 0) { //this wouldn't be ran since its being instantiated before
                            askQuestion(aVar);
                        }
                        clauseVarPointer.second += 1;
                    }
                    findTreatment(ruleNumber);
                    checkInstantiatedList(searchedIndex);
                    break;
                }
            }
        }
    }
}

void ForwardChaining::askQuestion(std::string missing) {
    if(missing == "HD") {
        std::cout << "Do you have a heart disease? (Yes/No): ";
        getline(std::cin, variableList[missing]);
        instantiatedList[missing] = 1;
    } else if(missing == "DN") {
        std::cout << "What disease do you have? ";
        getline(std::cin, variableList[missing]);
        instantiatedList[missing] = 1;
    }
}

std::string ForwardChaining::returnTreatment() {
    return variableList["TR"];
}

void ForwardChaining::findTreatment(int ruleNumber) {
    switch(ruleNumber) {
        case 10:
            if(variableList["HD"] == "No") {
                variableList["TR"] = "No treatment needed.";
            }
            break;
        case 20:
            if(variableList["HD"] == "Yes" && variableList["DN"] == "coronary artery disease") {
                variableList["TR"] = "Bypass Surgery";
            }
            break;
        case 30:
```

```
      if(variableList["HD"] == "Yes" && variableList["DN"] == "heart attack") {
        variableList["TR"] = "Bypass Surgery";
      }
      break;
    case 40:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "sudden cardiac arrest") {
        variableList["TR"] = "CPR";

      }
      break;
    case 50:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "cardiomegaly") {
        variableList["TR"] = "Medications";
      }
      break;
    case 60:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "arrhythmia") {
        variableList["TR"] = "Beta Blocker Medicine";
      }
      break;
    case 70:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "atrial fibrillation") {
        variableList["TR"] = "Cardioversion";
      }
      break;
    case 80:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "heart valve disease") {
        variableList["TR"] = "Heart Valve Replacement/Repair";
      }
      break;
    case 90:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "congenital heart disease") {
        variableList["TR"] = "Intracardiac repair surgery";
      }
      break;
    case 100:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "dilated cardiomyopathy") {
        variableList["TR"] = "Heart Transplant";
      }
      break;
    case 110:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "ischemic cardiomyopathy") {
        variableList["TR"] = "Cardiac Resynchronization Therapy";
      }
      break;
    case 120:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "restrictive cardiomyopathy") {
        variableList["TR"] = "Medications";
      }
      break;
    case 130:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "pericarditis") {
        variableList["TR"] = "Corticosteroids";
      }
      break;
    case 140:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "pericardial effusion") {
        variableList["TR"] = "Pericardiectomy";
      }
      break;
    case 150:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "marfan syndrome") {
        variableList["TR"] = "Medications";
      }
      break;
    case 160:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "heart murmurs") {
        variableList["TR"] = "Valve repair/replacment";
      }
      break;
    case 170:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "hypertrophic cardiomyopathy") {
        variableList["TR"] = "Septal Myectomy";
      }
      break;
    case 180:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "abdominal aortic aneurysm") {
        variableList["TR"] = "Endovascular repair";
      }
      break;
    case 190:
      if(variableList["HD"] == "Yes" && variableList["DN"] == "cardiomyopathy") {
        variableList["TR"] = "Surgery";
      }
      break;
     default:
      variableList["TR"] = "No treatment found.";
      break;
  }

  if(variableList["TR"] != "") {
    conclusionVarQueue.pop();
    conclusionVarQueue.push("TR");
    //cout << "Treatment = " << variableList["TR"] << endl;
  }
}
```

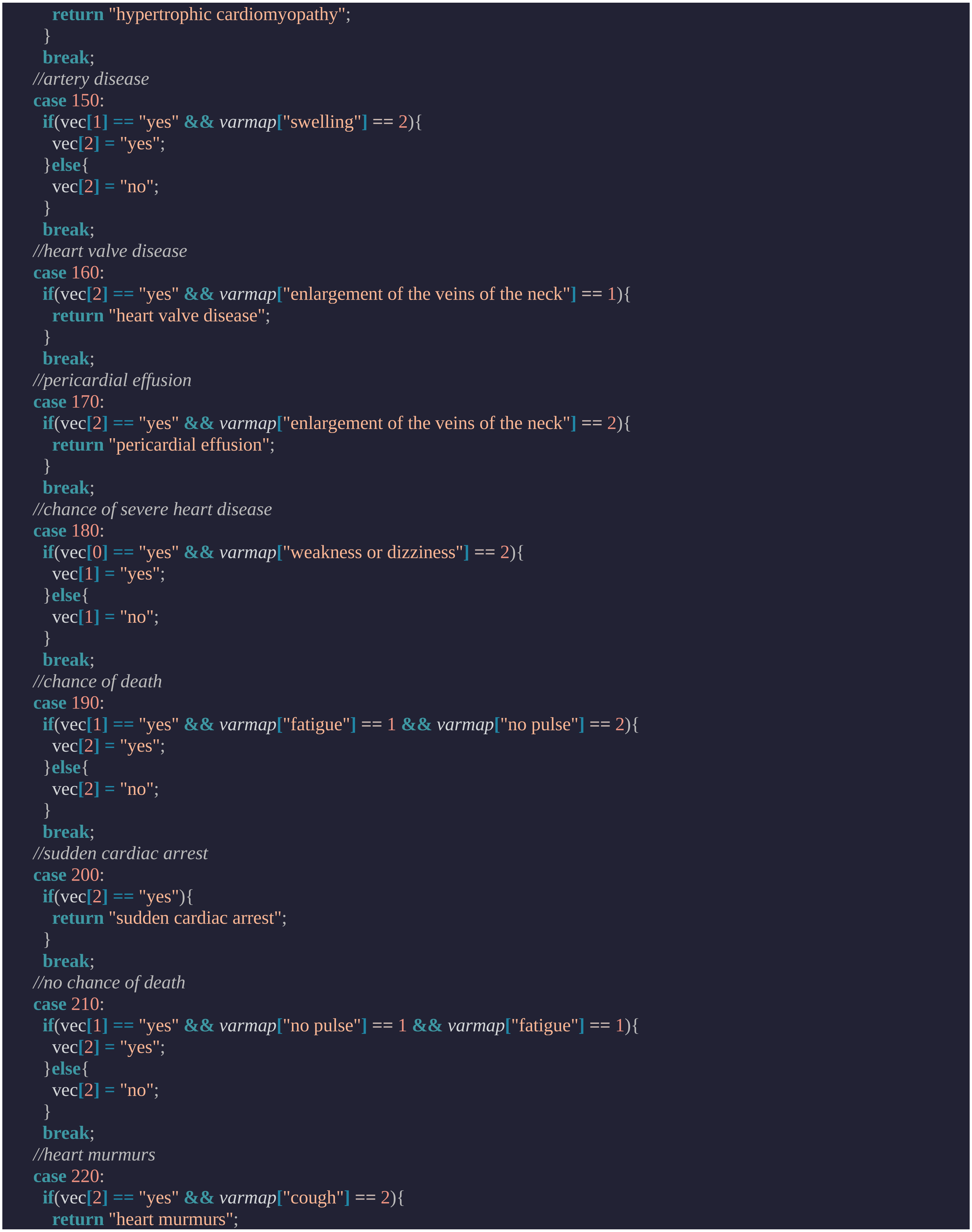
Main.cpp containing Backward chaining (main driver file)

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <unordered_map>
#include <stack>
#include <algorithm>
#include <utility>
#include <chrono>
#include "forward_chaining.h"

std::string rulesList(std::stack<std::pair<int, int>> &, std::unordered_map<std::string, int> &);
void searchClauseVar(std::stack<std::pair<int, int>> &,
                std::vector<std::pair<std::string, int>>,
                std::vector<std::string>,
                std::unordered_map<std::string, int> &);
void instVar(std::unordered_map<std::string, int> &, std::string);
std::vector<std::string> initClVarLt();
std::unordered_map<std::string, int> initVarLt();
std::vector<std::pair<std::string, int>> initConclT();
std::string bwInfEng(std::vector<std::pair<std::string, int>>,
                std::unordered_map<std::string, int> &,
                std::vector<std::string>);
// Test/debug Functions
// Helper Functions
bool isHeartDisease(std::pair<std::string, int>);
int findSubconclusion(std::vector<std::pair<std::string, int>>, std::string);

int main(){
  // start clock
  auto start = std::chrono::high_resolution_clock::now();
  //Init Variable List
  std::unordered_map<std::string, int> varmap = initVarLt();
  //Init Clause Variable List
  std::vector<std::string> clvarlt = initClVarLt();
  //Init Conclusion List
  std::vector<std::pair<std::string, int>> conclT = initConclT();
  std::string diagnosis = bwInfEng(conclT, varmap, clvarlt);
  std::cout << "Disease is: " << diagnosis << std::endl;

  std::string hasDisease = (diagnosis == "not heart disease") ? "No" : "Yes";
  ForwardChaining fw;
  fw.init();
  fw.provideDisease(hasDisease, diagnosis);
  std::string treatment = fw.returnTreatment();
  std::cout << "Treatment is: " << treatment << std::endl;

  // end clock, calc time, display
  auto stop = std::chrono::high_resolution_clock::now();
  auto duration = std::chrono::duration_cast<std::chrono::seconds>(stop - start);
  std::cout << "\nExecution time: " << duration.count() << " seconds." << std::endl;

  return 0;
}
```

```
std::string bwInfEng(std::vector<std::pair<std::string, int>> conclT,
                  std::unordered_map<std::string, int> &varmap,
                  std::vector<std::string> clvarLt) {
  std::string disease = "not heart disease";
  std::string cont;
  //conclusion stack has pair of ints, first int is conclusion number, 2nd int is clause number
  std::stack<std::pair<int, int>> conSt;
  //find heart disease
  auto it = std::find_if(conclT.begin(), conclT.end(), isHeartDisease);
  //push first instance of heart disease onto stack
  conSt.push(std::make_pair(it -> second, 4*((it -> second/10) - 1)));
  it++;

  std::cout << "------ Searching for heart disease ------\n";
  while((it != conclT.end() || !conSt.empty()) && disease == "not heart disease"){
    //instVar(conSt, conclT, clvarLt, varmap);
    searchClauseVar(conSt, conclT, clvarLt, varmap);
    //we have all the subconclusions pushed on to the stack, now we check the rules
    std::cout << "Checking Rules\n";
    disease = rulesList(conSt, varmap);
    if(disease == "not heart disease"){
      std::cout << "This is not the heart disease\n";
    }
    //if disease isn't found find next conclusion list
    if(it != conclT.end()){
      it = std::find_if(it, conclT.end(), isHeartDisease);
      conSt.push(std::make_pair(it -> second, 4*((it -> second/10) - 1)));
      it++;
    }
  }
  std::cout << "------ Found the heart disease ------\n";
  return disease;
}
void searchClauseVar(std::stack<std::pair<int, int>> &conSt,
                  std::vector<std::pair<std::string, int>> conclT,
                  std::vector<std::string> clvarLt,
                  std::unordered_map<std::string, int> &varmap) {
  //go through clause variable list
  int i = conSt.top().second;
  int j = i + 4;
  std::cout << "Instantiating Variables For Rule: " << conSt.top().first << "\n";
  for(; i != j; i++){
    //if the variable is not a sub conclusion
    if(clvarLt[i] == ""){
      break;
    }
    if(varmap.find(clvarLt[i]) != varmap.end()){
      // instVar
      if(varmap[clvarLt[i]] == 0){
        std::cout << "Clause Variable " << clvarLt[i] << " has not been instantiated\n";
        instVar(varmap, clvarLt[i]);
        //after we instantiate a variable check the rules
      }else{
        std::cout << "Clause Variable has been instantiated\n";
      }
    }else{
      std::cout << "Clause Variable is Subconclusion\n";
```

```
   int subclnum = findSubconclusion(conclT, clvarLt[i]);
   conSt.push(std::make_pair(subclnum, 4*(subclnum/10 - 1)));
   searchClauseVar(conSt, conclT, clvarLt, varmap);
  }
 }
}

std::string rulesList(std::stack<std::pair<int, int>> &conSt, std::unordered_map<std::string, int> &varmap){
 //we're now search through the rules
 //what are the conditions?
 //All the variables for a particular disease have been instantiated
 //lets check the rules
 //once we check the rules we should pop the stack
 //if the rule relies on sub conclusions we need a way to store that a rule has been check
 //we will keep check rules till the stack is empty
 std::vector<std::string> vec(4);
 while(!conSt.empty()){
  int num = conSt.top().first;
  conSt.pop();
  std::cout << "Checking Rules for " << "Rule Number: " << num << "\n";
  switch(num){
  //very low canche of heart disese
   case 10:
    if(varmap["shortness of breath"] == 2){
     vec[0] = "yes";
    }else if(varmap["shortness of breath"] == 1){
     vec[0] = "no";
    }
    break;
  //heart disease
   case 20:
    if(vec[0] == "no" && varmap["abdomen pain"] == 2){
     return "abdominal aortic aneurysm";
    }
    break;
  //genetic heart disease
   case 30:
    if( vec[0] == "no" && varmap["abdomen pain"] == 1){
     vec[1] = "yes";
    }else{
     vec[1] = "no";
    }
    break;
   //pericarditis
   case 40:
    if(vec[1] == "yes" && varmap["pain when swallowing"] == 2){
     return "pericarditis";
    }
    break;
   //Marfan syndrom
   case 50:
    if(vec[1] == "yes" && varmap["pain when swallowing"] == 1){
     return "marfan syndrome";
    }
    break;
   //less chance of heart disease
   case 60:
```

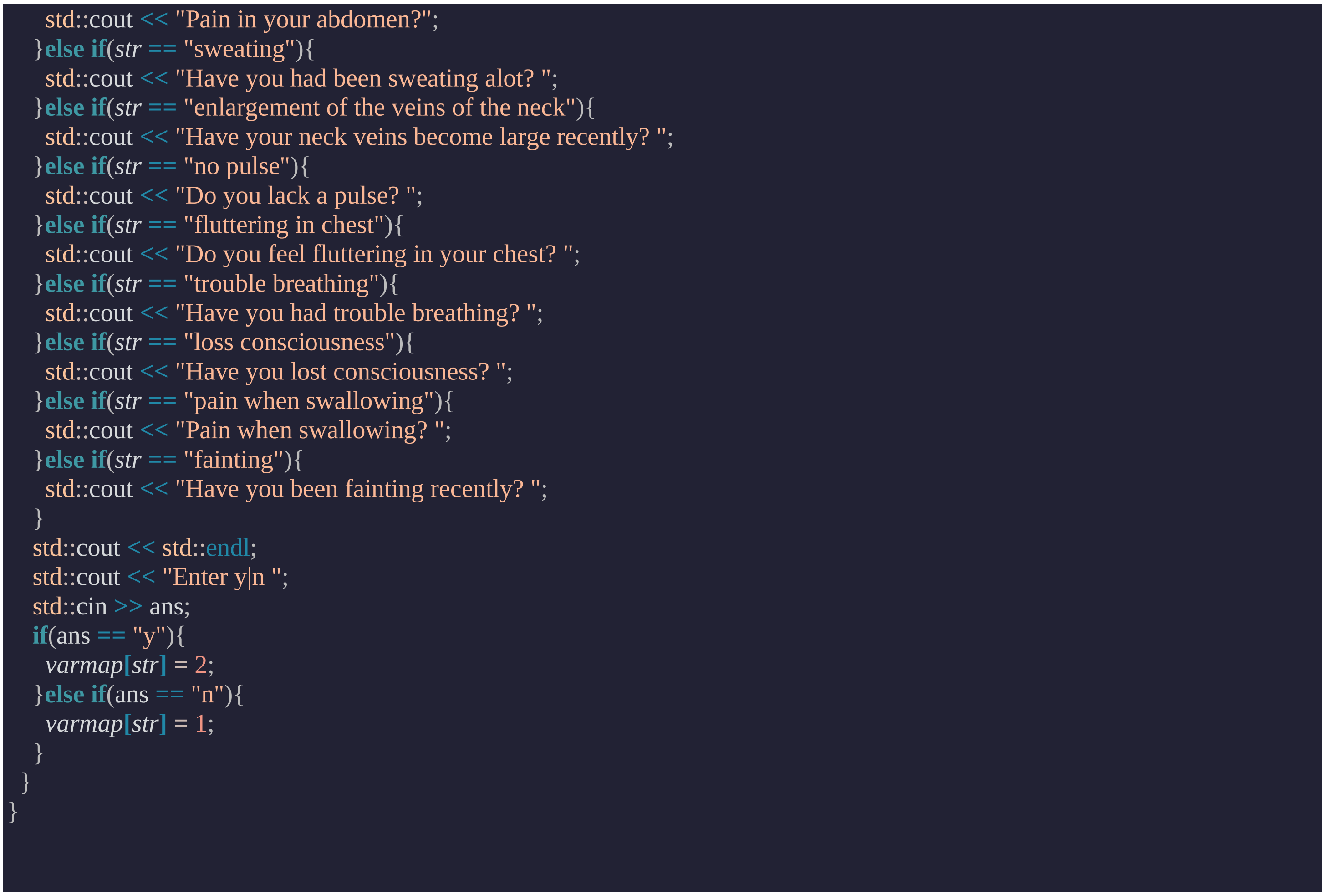
```
  if(varmap["shortness of breath"] == 2 && varmap["chest pain"] == 1){
    vec[0] = "yes";
  }else{
    vec[0] = "no";
  }
  break;
//congenital heart disease
case 70:
  if(vec[0] == "yes" && varmap["cough"] == 1){
    return "congenital heart disease";
  }
  break;
//some type of cardiomyopathy
case 80:
  if(vec[0] == "yes" && varmap["cough"] == 2){
    vec[1] = "yes";
  }else{
    vec[1] = "no";
  }
  break;
//dilated cardiomyopathy
case 90:
  if(vec[1] == "yes" && varmap["nausea"] == 1){
    return "dilated cardiomyopathy";
  }
  break;
//restrictive cardiomyopathy
case 100:
  if(vec[1] == "yes" && varmap["nausea"] == 2){
    return "restrictive cardiomyopathy";
  }
  break;
//chance of severe heart disease
case 110:
  if(varmap["shortness of breath"] == 2 && varmap["chest pain"] == 2){
    vec[0] = "yes";
  }else{
    vec[0] = "no";
  }
  break;
//coronary artery disease
case 120:
  if(vec[0] == "yes" && varmap["weakness or dizziness"] == 1 && varmap["fainting"] == 1){
    return "coronary artery disease";
  }
  break;
//advise going to the hospital
case 130:
  if(vec[0] == "yes" && varmap["weakness or dizziness"] == 1 && varmap["fainting"] == 2){
    vec[1] = "yes";
  }else{
    vec[1] = "no";
  }
  break;
//hypertrophic cardiomyopathy
case 140:
  if(vec[1] == "yes" && varmap["swelling"] == 1){
```



```
        return "hypertrophic cardiomyopathy";
      }
      break;
    //artery disease
    case 150:
      if(vec[1] == "yes" && varmap["swelling"] == 2){
        vec[2] = "yes";
      }else{
        vec[2] = "no";
      }
      break;
    //heart valve disease
    case 160:
      if(vec[2] == "yes" && varmap["enlargement of the veins of the neck"] == 1){
        return "heart valve disease";
      }
      break;
    //pericardial effusion
    case 170:
      if(vec[2] == "yes" && varmap["enlargement of the veins of the neck"] == 2){
        return "pericardial effusion";
      }
      break;
    //chance of severe heart disease
    case 180:
      if(vec[0] == "yes" && varmap["weakness or dizziness"] == 2){
        vec[1] = "yes";
      }else{
        vec[1] = "no";
      }
      break;
    //chance of death
    case 190:
      if(vec[1] == "yes" && varmap["fatigue"] == 1 && varmap["no pulse"] == 2){
        vec[2] = "yes";
      }else{
        vec[2] = "no";
      }
      break;
    //sudden cardiac arrest
    case 200:
      if(vec[2] == "yes"){
        return "sudden cardiac arrest";
      }
      break;
    //no chance of death
    case 210:
      if(vec[1] == "yes" && varmap["no pulse"] == 1 && varmap["fatigue"] == 1){
        vec[2] = "yes";
      }else{
        vec[2] = "no";
      }
      break;
    //heart murmurs
    case 220:
      if(vec[2] == "yes" && varmap["cough"] == 2){
        return "heart murmurs";
      }
      break;
    //cardiomyopathy
    case 230:
      if(vec[2] == "yes" && varmap["cough"] == 1){
        return "cardiomyopathy";
      }
      break;
    //atrial fibrillation
    case 240:
      if(vec[1] == "yes" && varmap["fatigue"] == 2 && varmap["sweating"] == 1 && varmap["palpitations"] == 2){
        return "atrial fibrillation";
      }
      break;
    //cardiomegaly
    case 250:
      if(vec[1] == "yes" && varmap["fatigue"] == 2 && varmap["sweating"] == 1 && varmap["palpitations"] == 1){
        return "cardiomegaly";
      }
      break;
    //emergency
    case 260:
      if(vec[1] == "yes" && varmap["fatigue"] == 2 && varmap["sweating"] == 2){
        vec[2] = "yes";
      }else{
        vec[2] = "no";
      }
      break;
    //require immediate attention
    case 270:
      if(vec[2] == "yes" && varmap["fluttering in chest"] == 1){
        vec[3] = "yes";
      }else{
        vec[3] = "no";
      }
      break;
    //heart attack
    case 280:
      if(vec[3] == "yes"){
        return "heart attack";
      }
      break;
    //doesn't require immediate attention
    case 290:
      if(vec[2] == "yes" && varmap["fluttering in chest"] == 2){
        vec[3] = "yes";
      }else{
        vec[3] = "no";
      }
      break;
    case 300:
      if(vec[3] == "yes"){
        return "arrhythmia";
      }
      break;
    default:
      break;
    }
  }
  return "not heart disease";
}


/* Initiate Conclusion Table
  Uses a hash map to map the name of the the goal/disease to its appropriate rule
  I.E. Heart Murmurs is mapped to the value of 10 so on and so forth
*/
std::vector<std::pair<std::string, int>> initConclT(){
  std::vector<std::pair<std::string, int>> conmap(30);
  conmap[0] = std::make_pair("very low chance of heart disease", 10);
  //abdominal aortic aneurysm
  conmap[1] = std::make_pair("heart disease", 20);
  conmap[2] = std::make_pair("genetic disease", 30);
  //pericarditis
  conmap[3] = std::make_pair("heart disease", 40);
  //Marfan Syndrome
  conmap[4] = std::make_pair("heart disease", 50);
  conmap[5] = std::make_pair("less chance of severe heart disease", 60);
  //congenital heart disease
  conmap[6] = std::make_pair("heart disease", 70);
  conmap[7] = std::make_pair("some type of cardiomyopathy", 80);
  //dilated cardiomyopathy
  conmap[8] = std::make_pair("heart disease", 90);
  //restrictive cardiomyopathy
  conmap[9] = std::make_pair("heart disease", 100);
  conmap[10] = std::make_pair("chance of severe heart disease", 110);
  //coronary artery disease
  conmap[11] = std::make_pair("heart disease", 120);
  conmap[12] = std::make_pair("advise going to hospital", 130);
  //hypertrophic cardiomyopathy
  conmap[13] = std::make_pair("heart disease", 140);
  conmap[14] = std::make_pair("artery disease", 150);
  //heart valve disease
  conmap[15] = std::make_pair("heart disease", 160);
  //pericardial effusion
  conmap[16] = std::make_pair("heart disease", 170);
  conmap[17] = std::make_pair("serious chance of severe heart disease", 180);
  conmap[18] = std::make_pair("chance of death", 190);
  conmap[19] = std::make_pair("heart disease", 200);
  conmap[20] = std::make_pair("no chance of death", 210);
  //heart murmurs
  conmap[21] = std::make_pair("heart disease", 220);
  //cardiomyopathy
  conmap[22] = std::make_pair("heart disease", 230);
  //atrial fibrillation
  conmap[23] = std::make_pair("heart disease", 240);
  //cardiomegaly
  conmap[24] = std::make_pair("heart disease", 250);
  conmap[25] = std::make_pair("emergency", 260);
  conmap[26] = std::make_pair("require immediate attention", 270);
  //heart attack
  conmap[27] = std::make_pair("heart disease", 280);
  conmap[28] = std::make_pair("doesn't require immediate attention", 290);
  //arrhythmia
  conmap[29] = std::make_pair("heart disease", 300);
  return conmap;
}

/*
  Init Varirable Table
  Uses Unordered Map
  If value == 0 then Variable in uninstantiated
  If value == 1 then variable is false
  If value == 2 then variable is true
*/
std::unordered_map<std::string, int> initVarLt(){
  std::unordered_map<std::string, int> varmap;
  varmap["cough"] = 0;
  varmap["swelling"] = 0;
  varmap["weakness or dizziness"] = 0;
  varmap["shortness of breath"] = 0;
  varmap["chest pain"] = 0;
  varmap["fatigue"] = 0;
  varmap["palpitations"] = 0;
  varmap["nausea"] = 0;
  varmap["abdomen pain"] = 0;
  varmap["sweating"] = 0;
  varmap["enlargement of the veins of the neck"] = 0;
  varmap["no pulse"] = 0;
  varmap["fluttering in chest"] = 0;
  varmap["trouble breathing"] = 0;
  varmap["loss consciousness"] = 0;
  varmap["pain when swallowing"] = 0;
  varmap["fainting"] = 0;
  return varmap;
}

std::vector<std::string> initClVarLt(){
  std::vector<std::string> clVarLt(120);
  for(auto &i : clVarLt){
    i = "";
  }
  //Very low chance of heart disease 10
  clVarLt[0] = "shortness of breath";
  //abdominal aortic aneurysm 20
  clVarLt[4] = "very low chance of heart disease";
  clVarLt[5] = "abdomen pain";
  //genetic disease 30
  clVarLt[8] = "very low chance of heart disease";
  clVarLt[9] = "abdomen pain";
  //pericarditis 40
  clVarLt[12] = "genetic disease";
  clVarLt[13] = "pain when swallowing";
  //marfan syndrome 50
  clVarLt[16] = "genetic disease";
  clVarLt[17] = "pain when swallowing";
  //less chance of severe heart disease 60
  clVarLt[20] = "shortness of breath";
  clVarLt[21] = "chest pain";
  //congenital heart disease 70
  clVarLt[24] = "less chance of severe heart disease";
```

```
clVarLt[25] = "cough";
//some type of cardiomyopathy 80
clVarLt[28] = "less chance of severe heart disease";
clVarLt[29] = "cough";
//dilated cardiomyopathy 90
clVarLt[32] = "some type of cardiomyopathy";
clVarLt[33] = "nausea";
//restrictive cardiomyopathy 100
clVarLt[36] = "some type of cardiomyopathy";
clVarLt[37] = "nausea";
//chance of severe heart disease 110
clVarLt[40] = "shortness of breath";
clVarLt[41] = "chest pain";
//coronary artery disease 120
clVarLt[44] = "chance of severe heart disease";
clVarLt[45] = "weakness or dizziness";
clVarLt[46] = "fainting";
//advise going to hospital 130
clVarLt[48] = "chance of severe heart disease";
clVarLt[49] = "weakness or dizziness";
clVarLt[50] = "fainting";
//hypertrophic cardiomyopathy 140
clVarLt[52] = "advise going to hospital";
clVarLt[53] = "swelling";
//artery disease 150
clVarLt[56] = "advise going to hospital";
clVarLt[57] = "swelling";
//heart valve disease 160
clVarLt[60] = "artery disease";
clVarLt[61] = "enlargement of the veins of the neck";
//pericardial effusion 170
clVarLt[64] = "artery disease";
clVarLt[65] = "enlargement of the veins of the neck";
//serious chance of severe heart disease 180
clVarLt[68] = "chance of severe heart disease";
clVarLt[69] = "weakness or dizziness";
//chance of death 190
clVarLt[72] = "serious chance of severe heart disease";
clVarLt[73] = "fatigue";
clVarLt[74] = "no pulse";
//sudden cardiac arrest 200
clVarLt[76] = "chance of death";
//no chance of death 210
clVarLt[80] = "serious chance of severe heart disease";
clVarLt[81] = "fatigue";
clVarLt[82] = "no pulse";
//heart murmurs 220
clVarLt[84] = "no chance of death";
clVarLt[85] = "cough";
//cardio myopathy 230
clVarLt[88] = "no chance of death";
clVarLt[89] = "cough";
//atrial fibrillation 240
clVarLt[92] = "serious chance of severe heart disease";
clVarLt[93] = "fatigue";
clVarLt[94] = "sweating";
clVarLt[95] = "palpitations";
```

```
  //cardiomegaly 250
  clVarLt[96] = "serious chance of severe heart disease";
  clVarLt[97] = "fatigue";
  clVarLt[98] = "sweating";
  clVarLt[99] = "palpitations";
  //emergency 260
  clVarLt[100] = "serious chance of severe heart disease";
  clVarLt[101] = "fatigue";
  clVarLt[102] = "sweating";
  // require immediate attention 270
  clVarLt[104] = "emergency";
  clVarLt[105] = "fluttering in chest";
  // heart attack 280
  clVarLt[108] = "require immediate attention";
  //doesn't require immediate attention 290
  clVarLt[112] = "emergency";
  clVarLt[113] = "fluttering in chest";
  //arrhythmia 300
  clVarLt[116] = "doesn't require immediate attention";
  return clVarLt;
}
//Debug Functions
//Helper Functions
bool isHeartDisease(std::pair<std::string, int> i){
  return i.first == "heart disease";
}

int findSubconclusion(std::vector<std::pair<std::string, int>> conclt, std::string str){
  for(auto i : conclt){
    if(i.first == str){
      return i.second;
    }
  }
  return -1;
}

void instVar(std::unordered_map<std::string, int> &varmap, std::string str){
  if(varmap[str] == 0){
    //instantiate variable
    std::string ans;
    if(str == "shortness of breath"){
      std::cout << "Do you have shortness of breath? ";
    }else if(str == "cough"){
      std::cout << "Are you coughing alot? ";
    }else if(str == "swelling"){
      std::cout << "Do you have swelling? ";
    }else if(str == "weakness or dizziness"){
      std::cout << "Have you been weak or dizzy? ";
    }else if(str == "chest pain"){
      std::cout << "Are you have chest pain? ";
    }else if(str == "fatigue"){
      std::cout << "Are you feeling fatigued? ";
    }else if(str == "palpitations"){
      std::cout << "Have you had palpitations? ";
    }else if(str == "nausea"){
      std::cout << "Have you been suffering from nausea? ";
    }else if(str == "abdomen pain"){
```



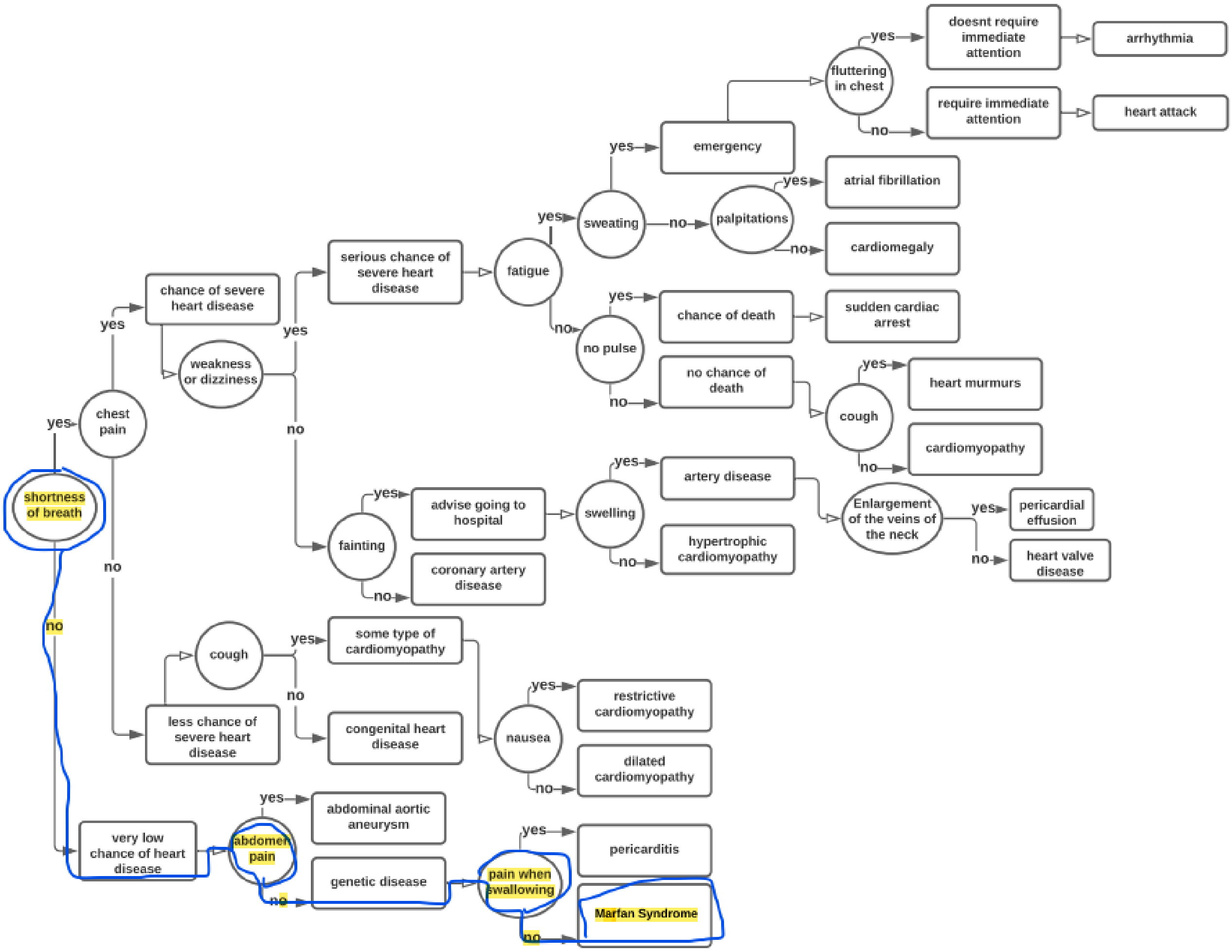
```
      std::cout << "Pain in your abdomen?";
    }else if(str == "sweating"){
      std::cout << "Have you had been sweating alot? ";
    }else if(str == "enlargement of the veins of the neck"){
      std::cout << "Have your neck veins become large recently? ";
    }else if(str == "no pulse"){
      std::cout << "Do you lack a pulse? ";
    }else if(str == "fluttering in chest"){
      std::cout << "Do you feel fluttering in your chest? ";
    }else if(str == "trouble breathing"){
      std::cout << "Have you had trouble breathing? ";
    }else if(str == "loss consciousness"){
      std::cout << "Have you lost consciousness? ";
    }else if(str == "pain when swallowing"){
      std::cout << "Pain when swallowing? ";
    }else if(str == "fainting"){
      std::cout << "Have you been fainting recently? ";
    }
    std::cout << std::endl;
    std::cout << "Enter y|n ";
    std::cin >> ans;
    if(ans == "y"){
      varmap[str] = 2;
    }else if(ans == "n"){
      varmap[str] = 1;
    }
  }
}
```

PROGRAM RUNS

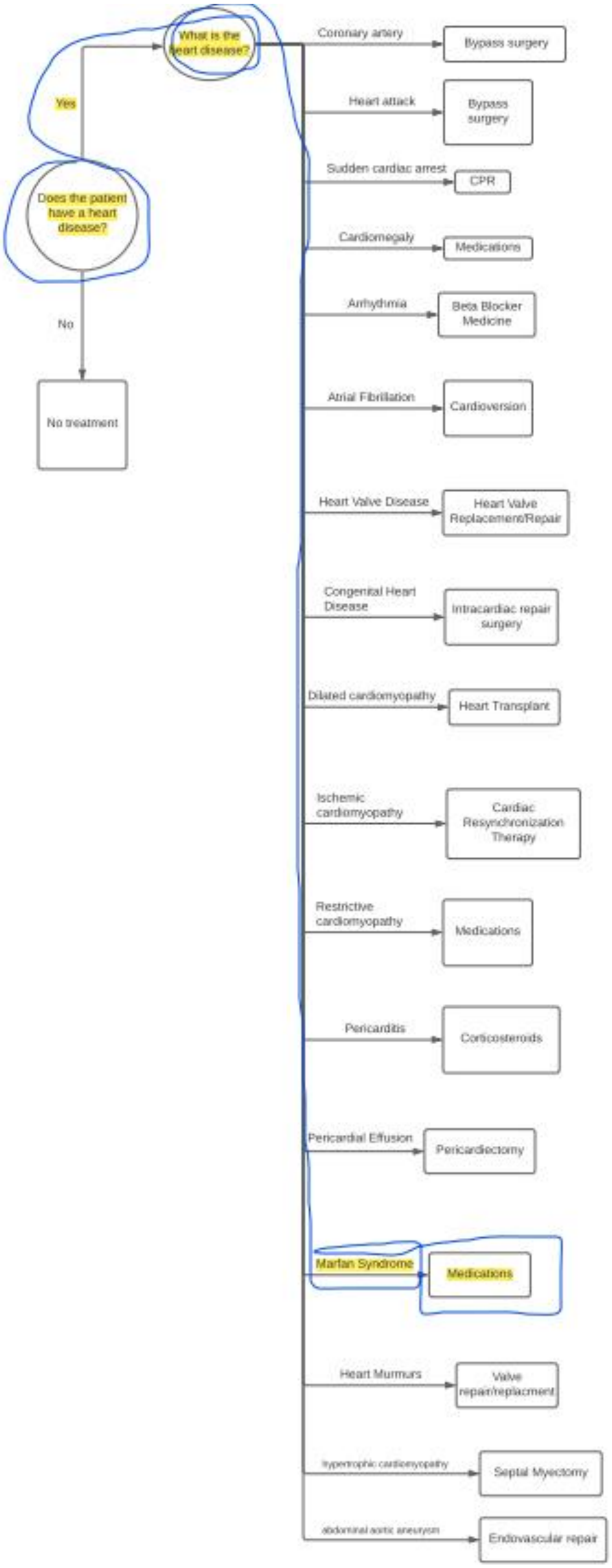
Run 1:

```
PS C:\Users\kody\Desktop\code\proj1\AICardiovascular\AICardiovascular\Project1> .\run.exe
------ Searching for heart disease ------
Instantiating Variables For Rule: 20
Clause Variable is Sub conclusion
Instantiating Variables For Rule: 10
Clause Variable shortness of breath has not been instantiated
Do you have shortness of breath?
Enter y|n n
Clause Variable abdomen pain has not been instantiated
Pain in your abdomen?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 20
This is not the heart disease
Instantiating Variables For Rule: 40
Clause Variable is Sub conclusion
Instantiating Variables For Rule: 30
Clause Variable is Sub conclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable pain when swallowing has not been instantiated
Pain when swallowing?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 40
This is not the heart disease
Instantiating Variables For Rule: 50
Clause Variable is Sub conclusion
Instantiating Variables For Rule: 30
Clause Variable is Sub conclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 50
------ Found the heart disease ------
Disease is: marfan syndrome
Treatment is: Medications
```

Run 1 example backward chaining tree path:



Run 1 forward chaining tree path:



Run 2:

```
C:\Users\kody\Desktop\code\proj1\AICardiovascular\AICardiovascular\Project1>run.exe
------ Searching for heart disease ------
Instantiating Variables For Rule: 20
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable shortness of breath has not been instantiated
Do you have shortness of breath?
Enter y|n y
Clause Variable abdomen pain has not been instantiated
Pain in your abdomen?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 20
This is not the heart disease
Instantiating Variables For Rule: 40
Clause Variable is Subconclusion
Instantiating Variables For Rule: 30
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable pain when swallowing has not been instantiated
Pain when swallowing?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 40
This is not the heart disease
Instantiating Variables For Rule: 50
Clause Variable is Subconclusion
Instantiating Variables For Rule: 30
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 50
This is not the heart disease
Instantiating Variables For Rule: 70
Clause Variable is Subconclusion
Instantiating Variables For Rule: 60
Clause Variable has been instantiated
Clause Variable chest pain has not been instantiated
Are you have chest pain?
Enter y|n y
```

```
Clause Variable cough has not been instantiated
Are you coughing alot?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 60
Checking Rules for Rule Number: 70
This is not the heart disease
Instantiating Variables For Rule: 90
Clause Variable is Subconclusion
Instantiating Variables For Rule: 80
Clause Variable is Subconclusion
Instantiating Variables For Rule: 60
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable nausea has not been instantiated
Have you been suffering from nausea?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 60
Checking Rules for Rule Number: 80
Checking Rules for Rule Number: 90
This is not the heart disease
Instantiating Variables For Rule: 100
Clause Variable is Subconclusion
Instantiating Variables For Rule: 80
Clause Variable is Subconclusion
Instantiating Variables For Rule: 60
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 60
Checking Rules for Rule Number: 80
Checking Rules for Rule Number: 100
This is not the heart disease
Instantiating Variables For Rule: 120
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable weakness or dizziness has not been instantiated
Have you been weak or dizzy?
Enter y|n y
Clause Variable fainting has not been instantiated
Have you been fainting recently?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 120
This is not the heart disease
```

```
Instantiating Variables For Rule: 140
Clause Variable is Subconclusion
Instantiating Variables For Rule: 130
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable swelling has not been instantiated
Do you have swelling?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 130
Checking Rules for Rule Number: 140
This is not the heart disease
Instantiating Variables For Rule: 160
Clause Variable is Subconclusion
Instantiating Variables For Rule: 150
Clause Variable is Subconclusion
Instantiating Variables For Rule: 130
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable enlargement of the veins of the neck has not been instantiated
Have your neck veins become large recently?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 130
Checking Rules for Rule Number: 150
Checking Rules for Rule Number: 160
This is not the heart disease
Instantiating Variables For Rule: 170
Clause Variable is Subconclusion
Instantiating Variables For Rule: 150
Clause Variable is Subconclusion
Instantiating Variables For Rule: 130
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 130
Checking Rules for Rule Number: 150
Checking Rules for Rule Number: 170
This is not the heart disease
Instantiating Variables For Rule: 200
Clause Variable is Subconclusion
Instantiating Variables For Rule: 190
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable fatigue has not been instantiated
Are you feeling fatigued?
Enter y|n n
Clause Variable no pulse has not been instantiated
Do you lack a pulse?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 190
Checking Rules for Rule Number: 200
This is not the heart disease
Instantiating Variables For Rule: 220
Clause Variable is Subconclusion
Instantiating Variables For Rule: 210
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 210
Checking Rules for Rule Number: 220
------ Found the heart disease ------
Disease is: cardiomyopathy
Treatment is: Surgery
```

Run 3:

```
C:\Users\kody\Desktop\code\proj1\AICardiovascular\AICardiovascular\Project1>run.exe
------ Searching for heart disease ------
Instantiating Variables For Rule: 20
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable shortness of breath has not been instantiated
Do you have shortness of breath?
Enter y|n n
Clause Variable abdomen pain has not been instantiated
Pain in your abdomen?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 20
------ Found the heart disease ------
Disease is: abdominal aortic aneurysm
Treatment is: Endovascular repair

Execution time: 3 seconds.
```

Run 4:

```
C:\Users\kody\Desktop\code\proj1\AICardiovascular\AICardiovascular\Project1>run.exe
------ Searching for heart disease ------
Instantiating Variables For Rule: 20
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable shortness of breath has not been instantiated
Do you have shortness of breath?
Enter y|n y
Clause Variable abdomen pain has not been instantiated
Pain in your abdomen?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 20
This is not the heart disease
Instantiating Variables For Rule: 40
Clause Variable is Subconclusion
Instantiating Variables For Rule: 30
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable pain when swallowing has not been instantiated
Pain when swallowing?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 40
This is not the heart disease
Instantiating Variables For Rule: 50
Clause Variable is Subconclusion
Instantiating Variables For Rule: 30
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 50
This is not the heart disease
Instantiating Variables For Rule: 70
Clause Variable is Subconclusion
Instantiating Variables For Rule: 60
Clause Variable has been instantiated
Clause Variable chest pain has not been instantiated
Are you have chest pain?
Enter y|n n
```

```
Clause Variable cough has not been
instantiated
Are you coughing alot?
Enter y|n n
Checking Rules
Checking Rules for Rule Number: 60
Checking Rules for Rule Number: 70
------ Found the heart disease ------
Disease is: congenital heart disease
Treatment is: Intracardiac repair surgery

Execution time: 5 seconds.
```

Run 5:

```
C:\Users\kody\Desktop\code\proj1\AICardiovascular\AICardiovascular\Project1>run.exe
------ Searching for heart disease ------
Instantiating Variables For Rule: 20
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable shortness of breath has not been instantiated
Do you have shortness of breath?
Enter y|n y
Clause Variable abdomen pain has not been instantiated
Pain in your abdomen?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 20
This is not the heart disease
Instantiating Variables For Rule: 40
Clause Variable is Subconclusion
Instantiating Variables For Rule: 30
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable pain when swallowing has not been instantiated
Pain when swallowing?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 40
This is not the heart disease
Instantiating Variables For Rule: 50
Clause Variable is Subconclusion
Instantiating Variables For Rule: 30
Clause Variable is Subconclusion
Instantiating Variables For Rule: 10
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 10
Checking Rules for Rule Number: 30
Checking Rules for Rule Number: 50
This is not the heart disease
Instantiating Variables For Rule: 70
Clause Variable is Subconclusion
Instantiating Variables For Rule: 60
Clause Variable has been instantiated
Clause Variable chest pain has not been instantiated
Are you have chest pain?
Enter y|n y
Clause Variable cough has not been instantiated
Are you coughing alot?
```

```
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 60
Checking Rules for Rule Number: 70
This is not the heart disease
Instantiating Variables For Rule: 90
Clause Variable is Subconclusion
Instantiating Variables For Rule: 80
Clause Variable is Subconclusion
Instantiating Variables For Rule: 60
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable nausea has not been instantiated
Have you been suffering from nausea?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 60
Checking Rules for Rule Number: 80
Checking Rules for Rule Number: 90
This is not the heart disease
Instantiating Variables For Rule: 100
Clause Variable is Subconclusion
Instantiating Variables For Rule: 80
Clause Variable is Subconclusion
Instantiating Variables For Rule: 60
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 60
Checking Rules for Rule Number: 80
Checking Rules for Rule Number: 100
This is not the heart disease
Instantiating Variables For Rule: 120
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable weakness or dizziness has not been instantiated
Have you been weak or dizzy?
Enter y|n y
Clause Variable fainting has not been instantiated
Have you been fainting recently?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 120
This is not the heart disease
Instantiating Variables For Rule: 140
Clause Variable is Subconclusion
Instantiating Variables For Rule: 130
```

```
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable swelling has not been instantiated
Do you have swelling?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 130
Checking Rules for Rule Number: 140
This is not the heart disease
Instantiating Variables For Rule: 160
Clause Variable is Subconclusion
Instantiating Variables For Rule: 150
Clause Variable is Subconclusion
Instantiating Variables For Rule: 130
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable enlargement of the veins of the neck has not been instantiated
Have your neck veins become large recently?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 130
Checking Rules for Rule Number: 150
Checking Rules for Rule Number: 160
This is not the heart disease
Instantiating Variables For Rule: 170
Clause Variable is Subconclusion
Instantiating Variables For Rule: 150
Clause Variable is Subconclusion
Instantiating Variables For Rule: 130
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 130
Checking Rules for Rule Number: 150
Checking Rules for Rule Number: 170
```

```
This is not the heart disease
Instantiating Variables For Rule: 200
Clause Variable is Subconclusion
Instantiating Variables For Rule: 190
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable fatigue has not been instantiated
Are you feeling fatigued?
Enter y|n y
Clause Variable no pulse has not been instantiated
Do you lack a pulse?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 190
Checking Rules for Rule Number: 200
This is not the heart disease
Instantiating Variables For Rule: 220
Clause Variable is Subconclusion
Instantiating Variables For Rule: 210
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 210
Checking Rules for Rule Number: 220
This is not the heart disease
Instantiating Variables For Rule: 230
Clause Variable is Subconclusion
Instantiating Variables For Rule: 210
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 210
Checking Rules for Rule Number: 230
This is not the heart disease
Instantiating Variables For Rule: 240
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable sweating has not been instantiated
Have you had been sweating alot?
Enter y|n y
Clause Variable palpitations has not been instantiated
Have you had palpitations?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 240
This is not the heart disease
Instantiating Variables For Rule: 250
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 250
This is not the heart disease
Instantiating Variables For Rule: 280
Clause Variable is Subconclusion
Instantiating Variables For Rule: 270
Clause Variable is Subconclusion
Instantiating Variables For Rule: 260
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
```

```
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable fluttering in chest has not been instantiated
Do you feel fluttering in your chest?
Enter y|n y
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 260
Checking Rules for Rule Number: 270
Checking Rules for Rule Number: 280
This is not the heart disease
Instantiating Variables For Rule: 300
Clause Variable is Subconclusion
Instantiating Variables For Rule: 290
Clause Variable is Subconclusion
Instantiating Variables For Rule: 260
Clause Variable is Subconclusion
Instantiating Variables For Rule: 180
Clause Variable is Subconclusion
Instantiating Variables For Rule: 110
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Clause Variable has been instantiated
Checking Rules
Checking Rules for Rule Number: 110
Checking Rules for Rule Number: 180
Checking Rules for Rule Number: 260
Checking Rules for Rule Number: 290
Checking Rules for Rule Number: 300
------ Found the heart disease ------
Disease is: arrhythmia
Treatment is: Beta Blocker Medicine

Execution time: 7 seconds.
```

ANALYSIS

Our program isn’t as efficient as its potential counterparts. Using recursion will always increase time complexity in programs. There are ways for us to increase efficiency, including: not traversing the entire decision tree on every run and increasing sub conclusions.

Our compile time is listed below at 467ms.

Our total temporary memory usuage at th longest run time was 1540KB

```
PS C:\Users\kody\Desktop\code\proj1\AICardiovascular\AICardiovascular\Project1

>Measure-Command {g++ -std=c++11 main.cpp forward_chaining.cpp | Out-Default}


     Days              : 0
     Hours             : 0
     Minutes           : 0
     Seconds           : 2
     Milliseconds      : 139
     Ticks             : 21397915
     TotalDays         : 2.47661053240741E-05
     TotalHours        : 0.000594386527777778
     TotalMinutes      : 0.0356631916666667
     TotalSeconds      : 2.1397915
     TotalMilliseconds : 2139.7915
```

CONCLUSION

The Advantages of Using Expert System

An expert system can be reliably used for many domains. The assistance provided from an expert system is undoubtedly essential and highly reliable to solve some solutions. Examples given below will be the advantages for the implementation of an expert system:

1. Providing consistent solutions – It can provide consistent answers for repetitive decisions, processes, and tasks. If the rule base in the system remains the same, regardless of how many times similar problems are being tested, the final conclusions drawn will remain the same.
2. Provides reasonable explanations – It can clarify the reasons why the conclusion was drawn and be why it is considered as the most logical choice among other alternatives.
3. Overcome human limitations – It does not have human limitations.
4. Easy to adapt to new conditions – An expert system has high adaptability and can meet new requirements in a short period of time. It also can capture new knowledge from an expert and use it as inference rules to solve new problems.

The Disadvantages of Using Expert System

Although the expert system does provide many significant advantages, it does have its drawbacks as well:

1. Lacks common sense – It lacks some decision making since all the decisions made are based on the inference rules set in the system. It also cannot make creative and innovative responses.
2. Difficulty in creating inference rules – Domain experts will not be able to always explain their logic and reasoning needed for the knowledge engineering process. May provide wrong solutions – There may be errors occurred in the processing due to some logic mistakes made in the knowledge base, which it will then provide the wrong solutions.

Summary

It is entirely subjective as to whether the advantages of expert system overweigh the disadvantages of implementing it. It depends on your domain and objectives. However, in my opinion, the implementation of expert system is critical in solving solutions using AI. Humans also have limitations as to how much knowledge a human is able to digest and comprehend. As for expert system, it can store as much knowledge as possible base on its storage space. Hence, in terms of performance, expert system is capable to perform as good if not better then human in specific instances.

REFERENCES

https://www.ukessays.com/essays/information-systems/the-expert-system.php

https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118

https://www.cdc.gov/heartdisease/about.htm

https://www.webmd.com/heart-disease

https://github.com/topics/expert-system

CONTRIBUTIONS

Levi: backward chaining program implementation and rules creation.

Tim: forward chaining program implementation, forward chaining tree and rule creation, linking backward and forward chaining files into a main driver.

Kody: backward chaining tree implementation, disease/symptom research, execution/compile time calculation, linking backward and forward chaining files into a main driver.