

INTELLIGENT EXPERT SYSTEM FOR DIAGNOSING AND TREATING CARDIOVASCULAR DISEASES

**Submitted by,
Gopika Mahadevan
Team members:
Sayali Pathak
Sri Sudha Kambhampati**

**Under the guidance of
Dr. Moonis Ali**

Contents

1.PROBLEM DESCRIPTION	3
2.THE DOMAIN	3
3.METHODOLOGIES	7
4.DECISION TREES	8
5.RULES.....	11
6.PROGRAM IMPLEMENTATION	22
7.SOURCE CODE.....	25
8.COPY OF PROGRAM RUNS	43
9.ANALYSIS OF THE PROGRAM.....	44
10.ANALYSIS OF THE RESULTS.....	44
11.CONCLUSION	45
12.REFERENCES	46
13.CONTRIBUTION	46

1.PROBLEM DESCRIPTION

1.1 DESCRIPTION OF THE PROBLEM

Cardiovascular diseases (CVDs) are a group of disorders of the heart and blood vessels. Cardiovascular diseases (CVDs) are the leading cause of death globally. An estimated 17.9 million people died from CVDs in 2019, representing 32% of all global deaths. Of these deaths, 85% were due to heart attack and stroke [1]. Early and on-time diagnosing of this problem is pivotal for preventing patients from more damage and saving their lives.

The major goal of our project is to create and deploy an intelligent expert system capable of diagnosing 30 such cardiovascular diseases. The user of this system can be a patient himself or a medical health practitioner. The working of the system has two parts, namely the diagnosing part and the treatment part.

For diagnosis the system will prompt the user give a yes/no or specific answer to questions about symptoms and based on the user's response the system will diagnose the disease. The system utilizes a backward chaining method to do this. A backward chaining decision tree is used for diagnosis, this tree captures rules to logically map the user's symptoms (received as input) to a CVD.

Once, the disease has been diagnosed, the systems will proceed to ask certain treatment specific questions. This part in the system utilizes the concept of forward chaining. A forward chaining tree takes in the disease name as input from the backward chaining and then analyses the answers to the treatment specific questions from the user to produce a set of treatment methods applicable to the patient/user as output.

This system has the potential to bring down the death caused by CVDs, because of its power to identify the cardiac diseases quickly and accurately. But it is important that it goes through multiple inspections and test case validations from health care professionals, doctors, and patients before its deployed for use in the real world. Provided the 30 cases work well the product can even be scaled up to diagnose many more such cardiac diseases.

1.2 OBJECTIVE OF THE PROJECT

Our objective is to efficiently and accurately diagnose diseases and recommend medications to the user by utilizing backward chaining methodology for diagnosis and forward chaining methodology for treatment recommendation. Ultimately, this approach aims to enhance the quality of healthcare by reducing the number of Cardiovascular related casualties and subsequently reducing the Cardiovascular death rate by providing prompt, accurate and tailored treatment recommendations for the patient.

2.THE DOMAIN

2.1 Intelligent Expert systems

This project has three core components a knowledge base, inference engine and an interface.

2.1.1 Knowledge Base

The knowledge base is a repository for domain-specific information, facts, rules, and heuristics used by the system to replicate the knowledge and competence of a human expert in a particular field. Knowledge for building our system required symptoms of 30 CVDs and their corresponding treatment methods. It is from this that we built rules, facts, and decision trees. This stored knowledge is processed by our inference engine.

2.1.2 Inference Engine

Inference engine processes knowledge to take decision. The inference engine relies heavily on rule-based reasoning to process the knowledge and rules in the knowledge base. We have used forward and backward chaining for providing diagnosis and treatment recommendations respectively:

Forward Chaining: In forward chaining, also known as data-driven reasoning, the inference engine starts with the available data or facts and applies rules to infer new conclusions.

Backward Chaining: In backward chaining, also known as goal-driven reasoning, the inference engine starts with a goal or hypothesis and works backward through the rules in the knowledge base to find the evidence or data that supports or refutes the goal.

2.1.3 Interface

We must ask user questions about the symptoms they might be having to diagnose a disease. And specific questions need to be asked to recommend medications. Interaction with the user is required, for this we are currently using the terminal.

2.2 Knowledge base required to diagnose and treat CVDs

As mentioned before accurate and quick diagnosis is pivotal to save a patient with a CVD. The huge amount of data available about CVDs on the internet has been used to build the intelligent system's knowledge base. We shortlisted 30 CVDs and collected symptoms and treatments of all such diseases. We had to further refine all the collected data to identify the symptoms unique to each CVD. These set of unique symptoms helps the system differentiate one CVD from another. This is a crucial step as many CVDs have similar symptoms.

The 30 CVDs that our system can detect are:

1. Ventricular Tachycardia
Symptoms: Chest pain, shortness of breath, fainting, neck tightness, heart palpitation, cardiac arrest
2. Takotsubo Cardiomyopathy (Broken Heart Syndrome)
Symptoms: Chest pain, fatigue, shortness of breath, weakening of left ventricle of heart, low blood pressure, heart palpitation
3. Long QT Syndrome
Symptoms: Chest pain, fatigue, shortness of breath, fainting, noisy gasping, seizures
4. Persistent atrial fibrillation
Symptoms: Chest pain, fatigue, shortness of breath all the time, dizziness, irregular heartbeat sometimes, nausea, heart palpitation

5. Pulmonary Hypertension
Symptoms: Chest pain, fatigue, never have shortness of breath, swelling, blue skin, dizziness, increased abdominal size
6. Aortic Aneurysm
Symptoms: Chest pain, fatigue, shortness of breath all the time, back pain, cough, scotch voice
7. Valvular heart disease
Symptoms: Chest pain, fatigue, shortness of breath all the time, dizziness, irregular heart beat all the time
8. Acute Coronary Syndrome
Symptoms: have coronary artery disease, nausea, indigestion, fainting, heavy sweating
9. Coronary Artery Disease
Symptoms: Chest Pain, fatigue, shortness of breath all the time, shoulder pain, heart palpitation, neck jaw pain
10. Rheumatic Heart Disease
Symptoms: Chest Pain, fatigue, shortness of breath all the time, fever, swollen joints , uncontrolled movement, skin lumps
11. Venous Thromboembolism
Symptoms: have pulmonary embolism, red limbs, tenderness of thigh, warm skin
12. Pulmonary Embolism
Symptoms: Chest Pain, fatigue, shortness of breath all the time, fever, heart murmur, blood in cough, discoloured skin
13. Endocarditis
Symptoms: Chest Pain, fatigue, shortness of breath all the time, fever, heart murmur, skin bumps
14. Ischemic Cardiomyopathy
Symptoms: Have arrhythmia, swelling weight gain
15. Arrhythmia
Symptoms: Chest Pain, fatigue, shortness of breath while lying down, blurry vision, heart palpitation
16. Heat Valve Stenosis
Symptoms: Chest Pain, fatigue, shortness of breath while lying down, difficulty in walking, swelling, prefer sitting
17. Brugada Syndrome

Symptoms: Chest Pain, fatigue, shortness of breath while lying down, fainting, irregular heartbeat, heart palpitation, seizures

18. Cardiac Tamponade

Symptoms, have pericarditis, infection or wound, cancer

19. Peripheral Artery Disease

Symptom: : Chest Pain, fatigue, shortness of breath while lying down, fever, sweaty chills, heart palpitation, dizziness

20. Coronary Microvascular Disease

Symptoms: Chest Pain, fatigue, shortness of breath during physical activity, sleep problem, lack of energy

21. Chronic Total Occlusion

Symptoms: Chest Pain, fatigue, shortness of breath during physical activity, dizziness, irregular heartbeat, nausea, heart palpitation, upper arm pain

22. Atherosclerosis

Symptoms: Chest Pain, fatigue, shortness of breath during physical activity, fainting, slurred speech, leg cramp, vision loss in one eye

23. Hypertrophic Cardiomyopathy

Symptoms: Chest Pain, fatigue, shortness of breath during physical activity, fainting dizziness, heart murmur, swelling

24. Mitral Valve Prolapse

Symptoms: Chest Pain, fatigue, shortness of breath during physical activity, fainting dizziness, heart murmur, swelling, numbness, history of cardiac arrest, history of Mitral Valve Prolapse

25. Hypertrophic Obstructive Cardio Myopathy

Symptoms: Chest Pain, fatigue, shortness of breath during physical activity, fainting dizziness, heart murmur, swelling, numbness, history of cardiac arrest

26. Eisenmenger Syndrome

Symptoms: Chest Pain, fatigue, shortness of breath during physical activity, fainting dizziness, heart murmur, blood in cough, blue color skin, heart palpitation

27. Cardiogenic Shock

Symptoms: Myocardial Infarction, less urine, pale skin

28. Myocardial Infarction

Symptoms: Chest pain, fatigue, shortness of breath, sweat chills, body pain

29. Wolff-Parkinson-White Syndrome

Symptoms: Chest pain, fatigue, shortness of breath with episodes of rapid heart rate, fainting, dizziness, heart palpitation, blood in cough, anxiety

30. Aortic Stenosis

Symptoms: Chest pain, fatigue, shortness of breath during physical activity, fainting, dizziness, heart murmur, blood in cough, sleep problem, swelling

3.METHODOLOGIES

3.1 Backward Chaining

This is a goal driven method. The goal in our system is to diagnose a CVD. In order to do this the program/inference engine matches the symptom entered with rules present in its knowledge base. Knowledge base of the system consists of rules. Rules are of the format:

if <symptom1> and <symptom2> and...then <disease1>

Here the if side is called the premise of the rule and then is called the conclusion of the rule. Many such rules are stored in the system.

If each clause in the premise side of any rule is satisfied the disease found in the conclusion side is taken as the output. That is the disease will be diagnosed. For example, the system proceeds to diagnose the disease with the following steps:

1. Checks if having a fever is a symptom of the disease. If the knowledge base confirms this, it moves to the next step.
2. Next, it checks if chest pain is a symptom of the disease. If the knowledge base confirms this, it moves to the next step.
3. Similarly, it checks if shortness of breath is a symptom of the disease. If the knowledge base confirms this, it moves to the next step.
4. Once all relevant symptoms have been checked, the system determines if the combination of symptoms observed in the patient is consistent with the disease. If the conclusion is affirmative based on the rules, it indicates that the patient likely has the disease. Otherwise, it concludes that the disease is unlikely.

3.2 Forward Chaining

This fact-driven method is used to recommend treatments based on the disease that was diagnosed by the backward chaining methodology. Knowledge base here too consists of rules. Rules are of the format:

if <symptom1> and <symptom2> and...then <list of medications>

The system proceeds by collecting various facts like does the person have lung problem then a bronchodilator is included in the medications, does he smoke then medications will include a warning to stop smoking. In forward chaining with facts a particular medication conclusion is derived.

4.DECISION TREES

Decision tree is a diagram which helps to visualize all the factors that must be considered for taking a decision. Decision trees have decision nodes with questions which branch out to further nodes, and finally reach a conclusion node. Working of a human mind while taking a decision is mimicked here. In our system, we have used a backward chaining and forward chaining tree to mimic this decision-making process.

4.1 Features of Backward and Forward chaining trees

- Decision nodes: Nodes which are oval and represents question asked to the user, expecting a yes/no answer.
 - In our backward chaining tree, each decision node holds a CVD symptom.
 - In the forward chaining tree, additional questions required to recommend treatment are represented using decision nodes.
- Links/Branches: Yes/no input from the user at a decision node passes control from the current decision node to the next node(next point of decision).
 - In our backward chaining tree, certain categorical splits have also been used. For example, shortness of breath can be all the time, while lying down, during physical activity, sudden, during episodes of rapid heart rate or never. Since cardiac diseases have very similar symptoms it is easier to categorize them based on such categorical variables. This makes the system's diagnosis accurate.
 - In forward chaining yes from the user passes control to next node. Categorical splitting has not been used.
- Final Nodes/Intermediate Nodes: Nodes which are rectangular and holds the result.
 - These nodes will have name of the diagnosed disease in the backward chaining tree.
 - Final nodes will have list of recommended treatments in forward chaining tree.

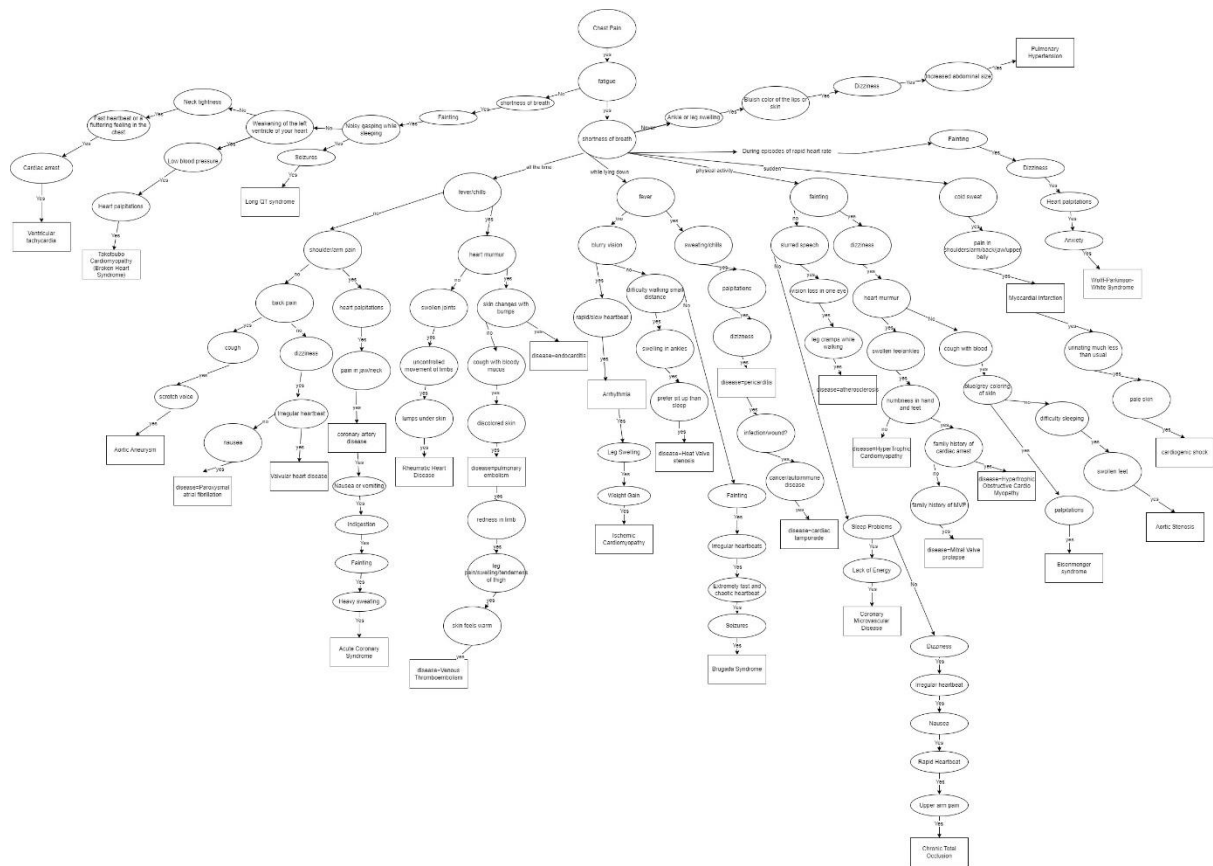


Fig 1.0: Backward chaining tree that we have designed for our system

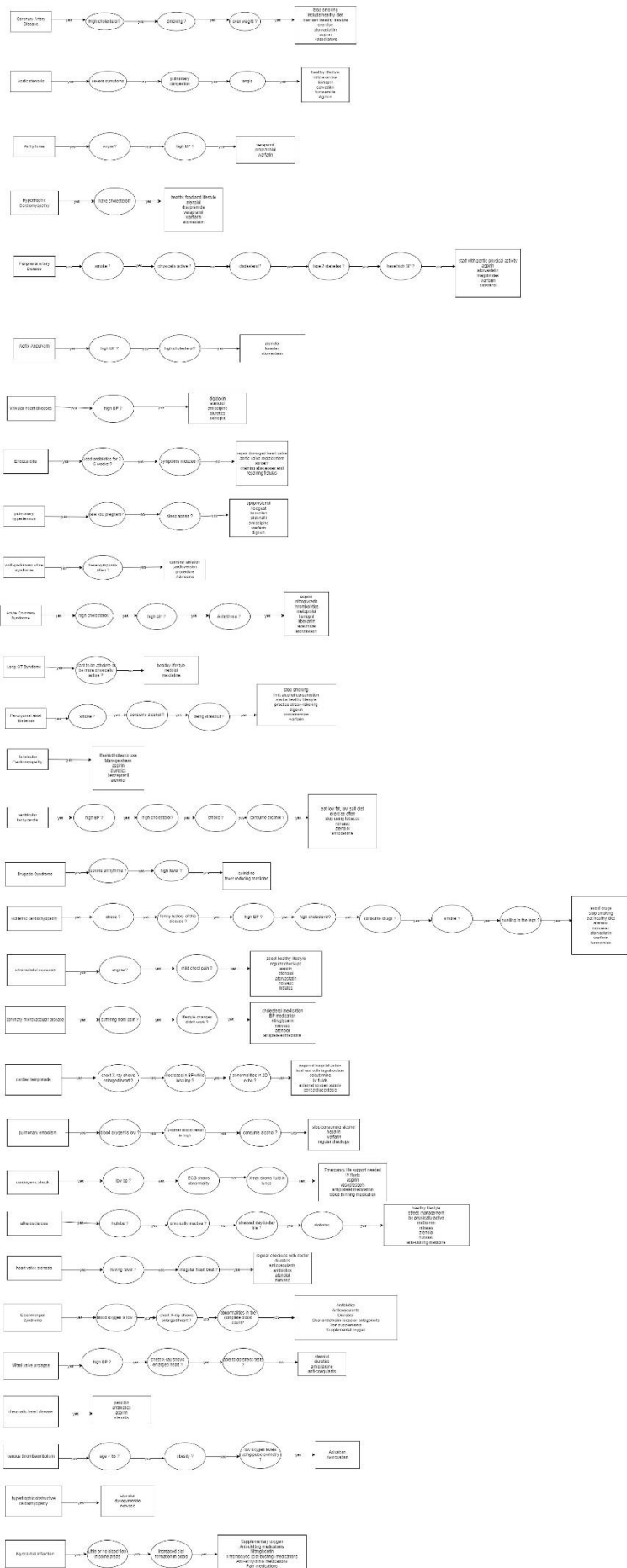


Fig 2.0: Forward chaining tree that we have designed for our system

The figures above have also been included as an attachment in case they are not clear here.

5.RULES

Rules consist of predefined conditions organized as "if-then" statements, where multiple clauses are linked by conjunctions to yield an apt decision. In our system, the CVD symptoms form the clauses in the if part, and the final decision value (name of the disease) is stored in the DIS (acronym we have used for disease) variable.

5.1 Backward chaining inference rules

5. IF CHESTPAIN = YES AND FATIGUE = NO AND SHORTNESSOFBREATH = YES AND FAINTING = YES AND NOISYGASPING = NO AND WEAKENINGINLEFTVENTRICLE = NO AND NECKTIGHTNESS = YES AND HEARTPALPITATION = YES AND CARDIACARST = YES THEN DIS = VENTRICULAR TACHYCARDIA
6. IF CHESTPAIN = YES AND FATIGUE = NO AND SHORTNESSOFBREATH = YES AND FAINTING = YES AND NOISYGASPING = NO AND WEAKENINGINLEFTVENTRICLE = YES AND LOWBP = YES AND HEARTPALPITATION = YES THEN DIS = TAKOTSUBO CARDIOMYOPATHY (BROKEN HEART SYNDROME)
7. IF CHESTPAIN = YES AND FATIGUE = NO AND SHORTNESSOFBREATH = YES AND FAINTING = YES AND NOISYGASPING = YES AND SEIZURES = YES THEN DIS = LONG QT SYNDROME
8. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ALL TIME AND FEVER = NO AND SHOULDERPAIN = NO AND BACKPAIN = NO AND DIZZINESS = YES AND IRREGULARHEARTBEAT = SOMETIMES AND NAUSEA = YES AND HEARTPALPITATION = YES THEN DIS = PAROXYSMAL ATRIAL FIBRILLATION
9. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = NEVER AND SWELLING = YES AND BLUESKIN = YES AND DIZZINESS = YES AND INCREASEDABDOMINALSIZE = YES THEN DIS = PULMONARY HYPERTENSION
10. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ALL TIME AND FEVER = NO AND SHOULDERPAIN = NO AND BACKPAIN = YES AND COUGH = YES AND SCRTHVOICE = YES THEN DIS = AORTIC ANEURSYM

11. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ALL TIME AND FEVER = NO AND SHOULDERPAIN = NO AND BACKPAIN = NO AND DIZZINESS = YES AND IRREGULARHEARTBEAT = ALLTIME THEN DIS = VALVULAR HEART DISEASE
12. IF CORONARY ARTERY DISEASE = YES AND NAUSEA = YES AND INDIGESTION = YES AND FAINTING = YES AND HEAVYSWEATING = YES THEN DIS = ACUTE CORONARY SYNDROME
13. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ALL TIME AND FEVER = NO AND SHOULDERPAIN = YES AND HEARTPALPITATION = YES AND NECKJWPAIN = YES THEN DIS = CORONARY ARTERY DISEASE
14. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ALL TIME AND FEVER = YES AND HEARTMURMUR = NO AND SWOLLENJOINTS = YES AND UNCONTROLLEDMOVEMENT = YES AND SKINLUMPS = YES THEN DIS = RHEUMATIC HEART DISEASE
15. IF PULMONARY EMBOLISM = YES AND REDLIMB = YES AND TENDERNESSOFTHIGH = YES AND WARMSKIN = YES THEN DIS = VENOUS THROMBOEMBOLISM
16. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ALL TIME AND FEVER = YES AND HEARTMURMUR = YES AND SKINBUMPS = NO AND BLOODINCOUGH = YES AND DISCOLOREDSKIN = YES THEN DIS = PULMONARY EMBOLISM
17. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ALL TIME AND FEVER = YES AND HEARTMURMUR = YES AND SKINBUMPS = YES THEN DIS = ENDOCARDITIS
18. IF ARRHYTHMIA = YES AND SWELLING= YES AND WEIGHTGAIN = YES THEN DIS = ISCHEMIC CARDIOMYOPATHY
19. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = LYINGDOWN AND FEVER = NO AND BLRYVISION = YES AND HEARTPALPITATION = YES THEN DIS = ARRHYTHMIA
20. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = LYINGDOWN AND FEVER = NO AND BLRYVISION = NO AND DIFFICULTYINWALKING = YES AND SWELLING = YES AND PREFERSSITTING = YES THEN DIS = HEAT VALVE STENOSIS

21. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = LYINGDOWN AND FEVER = NO AND BLRYVISION = NO AND DIFFICULTYINWALKING = NO AND FAINTING = YES AND IRREGULARHEARTBEAT = YES AND HEARTPALPITATION = YES AND SEIZURES = YES
THEN DIS = BRUGADA SYNDROME
22. IF PERICARDITIS = YES AND INFECTIONORWOUND = YES AND CANCER = YES
THEN DIS = CARDIAC TAMPONADE
23. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = LYINGDOWN AND FEVER = YES AND SWTCHILL = YES AND HEARTPALPITATION = YES AND DIZZINESS = YES
THEN DIS = PERIPHERAL ARTERY DISEASE
24. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ACTIVITY AND FAINTING = NO AND SLURRED SPEECH = NO AND SLEEP PROBLEM = YES AND LACK OF ENERGY = YES
THEN DIS = CORONARY MICROVASCULAR DISEASE
25. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ACTIVITY AND FAINTING = NO AND SLURRED SPEECH = NO AND SLEEP PROBLEM = NO AND DIZZINESS = YES AND IRREGULARHEARTBEAT = YES AND NAUSEA = YES AND HEARTPALPITATION = YES AND UPRARMPAIN = YES
THEN DIS = CHRONIC TOTAL OCCLUSION
26. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ACTIVITY AND FAINTING = NO AND SLURRED SPEECH = YES AND VSNLOSSONEI = YES AND LEGCRMP = YES
THEN DIS = ATHEROSCLEROSIS
27. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ACTIVITY AND FAINTING = YES AND DIZZINESS = YES AND HEARTMURMUR = YES AND SWELLING = YES AND NMBNESS = NO
THEN DIS = HYPERTROPHIC CARDIOMYOPATHY
28. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH = ACTIVITY AND FAINTING = YES AND DIZZINESS = YES AND HEARTMURMUR = YES AND SWELLING = YES AND NMBNESS = YES AND HSTRYCA = NO AND HSTRYMVP = YES
THEN DIS = MITRAL VALVE PROLAPSE

29. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH =
ACTIVITY AND FAINTING = YES AND DIZZINESS = YES AND HEARTMURMUR
= YES AND SWELLING = YES AND NMBNESS = YES AND HSTRYCA= YES
THEN DIS = HYPERTROPHIC OBSTRUCTIVE CARDIO MYOPATHY
30. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH =
ACTIVITY AND FAINTING = YES AND DIZZINESS = YES AND HEARTMURMUR
= NO AND BLOODINCOUGH = YES AND BLUECOLORSKIN = YES AND
HEARTPALPITATION = YES
THEN DIS = EISENMENGER SYNDROME
31. IF MYOCINFCTN = YES AND LESSURIN = YES AND PALESKN= YES
THEN DIS = CARDIOGENIC SHOCK
32. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH =
SUDDEN AND SWTCHILL = YES AND BDYPAIN = YES
THEN DIS = MYOCARDIAL INFARCTION
33. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH =
DURINGEPISODESOFRAPIDHEARTRATE AND FAINTING = YES AND
DIZZINESS = YES AND HEARTPALPITATION = YES AND ANXIETY = YES
THEN DIS = WOLFF-PARKINSON-WHITE SYNDROME
34. IF CHESTPAIN = YES AND FATIGUE = YES AND SHORTNESSOFBREATH =
ACTIVITY AND FAINTING = YES AND DIZZINESS = YES AND HEARTMURMUR
= NO AND BLOODINCOUGH = YES AND SLEEP PROBLEM = YES AND
SWELLING = YES
THEN DIS = AORTIC STENOSIS

5.2 Forward chaining inference rules

1. IF CORONARY ARTERY DISEASE=YES AND HIGH CHOLESTEROL?=YES AND
SMOKING?= YES AND OVER WEIGHT?= YES

THEN TREATMENT =

STOP SMOKING

INCLUDE HEALTHY DIET

MAINTAIN HEALTHY LIFESTYLE

EXERCISE

ATORVASTATTIN

ASPIRIN

VASODILATORS
2. IF AORTIC STENOSIS=YES AND SEVERE SYMPTOMS=NO AND PULMONARY
CONGESTION=YES AND ANGINA?=YES

THEN TREATMENT =
HEALTHY LIFESTYLE
MILD EXERCISE
LISINOPRIL
CARVEDILOL
FUROSEMIDE
DIGOXIN

3. IF ARRHYTHMIA?=YES AND ANGINA?= YES AND HIGH BP?=YES

THEN TREATMENT =
VERAPAMIL
PROPRANOLOL
WARFARIN

4. IF HYPERTROPHIC CARDIOMYOPATHY=YES AND HIGH CHOLESTEROL?=YES

THEN TREATMENT =
HEALTHY FOOD AND LIFESTYLE
ATENOLOL
DISOPRAMIDE
VERAPRAMIL
WARFIARIN
ATORVASTATIN

5. IF PERIPHERAL ARTERY DISEASE=YES AND SMOKING?=YES AND PHYSICALLY ACTIVE?=NO AND HIGH CHOLESTEROL?=YES AND TYPE 2 DIABETES?=YES AND HAVE HIGH BP?=YES

THEN TREATMENT =
START WITH GENTLE PHYSICAL ACTIVITY
ASPIRIN
ATORVASTATIN
MEGLITINIDES
WARFARIN
CILOSTAZOL

6. IF AORTIC ANEURYSM=YES AND HIGH BP?=YES AND HIGH CHOLESTEROL?=YES

THEN TREATMENT =

ATENOLOL

LOSARTAN

ATORVASTATIN

7. IF VALVULAR HEART DISEASES=YES AND HIGH BP?=YES

THEN TREATMENT =

DIGIDOXIN

ATENOLOL

AMLODIPINE

DIURETICS

LISINOPRIL

8. IF ENDOCARDITIS=YES AND USED ANTIBIOTICS FOR 2-6 WEEKS ?=YES AND SYMPTOMS REDUCED ?=NO

THEN TREATMENT =

REPAIR DAMAGED HEART VALVE

AORTIC VALVE REPLACEMENT SURGERY

DRAINING ABSCESES AND REPAIRING FISTULAS

9. IF PULMONARY HYPERTENSION=YES AND ARE YOU PREGNANT?=NO AND SLEEP APNEA ?=YES

THEN TREATMENT =

EPOPROSTENOL

RIOCIGUAT

BOSENTAN

SILDENAFIL

AMLODIPINE

WARFARIN

DIGOXIN

10. IF WOLFF-PARKINSON WHITE SYNDROME=YES AND HAVE SYMPTOMS OFTEN?=YES

THEN TREATMENT =

CATHERER ABLATION

CARDIOVERSION PROCEDURE

ADINOSINE

11. IF ACUTE CORONARY SYNDROME=YES AND HIGH CHOLESTEROL?= YES
AND HIGH BP?= YES AND ARRHYTHMIA?= YES

THEN TREATMENT =

ASPIRIN

NITROGLYCERIN

THROMBOLYTICS

METOPROLOL

LISINOPRIL

IRBESARTIN

EZETIMIBE

ATORVASTATIN

12. IF LONG QT SYNDROME=YES AND WANT TO BE ATHLETE OR BE MORE
PHYSICALLY ACTIVE ?=NO

THEN TREATMENT =

HEALTHY LIFESTYLE

NADOLOL

MEXILETINE

13. IF PAROXYSMAL ATRIAL FIBRILLATION=YES AND SMOKING?= YES AND
CONSUME ALCOHOL ?=YES AND BEING STRESSFUL ? = YES

THEN TREATMENT =

STOP SMOKING

LIMIT ALCOHOL CONSUMPTION

START A HEALTHY LIFESTYLE

PRACTICE STRESS-RELIEVING

DIGOXIN

PROCAINAMIDE

WARFARIN

14. IF TAKOTSUBO CARDIOMYOPATHY=YES

THEN TREATMENT =

RESTRICT TOBACCO USE

MANAGE STRESS

ASPIRIN

DIURETICS

BENZAPRAZIL

ATENOLOL

15. IF VENTRICULAR TACHYCARDIA=YES AND HIGH BP?= YES AND HIGH CHOLESTEROL?=YES AND SMOKING?= YES AND CONSUME ALCOHOL ?=YES

THEN TREATMENT =

EAT LOW FAT, LOW SALT DIET

EXERCISE OFTEN

STOP USING TOBACCO

NORVASC

ATENOLOL

AMIODARONE

16. IF BRUGADA SYNDROME=YES AND ARRHYTHMIA?=YES AND HIGH FEVER ? = YES

THEN TREATMENT =

QUINIDINE

FEVER REDUCING MEDICINE

17. IF ISCHEMIC CARDIOMYOPATHY=YES AND OBESE ? = YES AND FAMILY HISTORY OF THIS DISEASE ? = YES AND HIGH BP?= YES AND HIGH CHOLESTEROL?= YES AND CONSUME DRUGS ? = YES AND SMOKING?= YES AND SWELLING IN THE LEGS ? = YES

THEN TREATMENT =

AVOID DRUGS

STOP SMOKING

EAT HEALTHY DIET

ATENOLOL

NORVASAC

ATORVASTATIN

WARFARIN

FUROSEMIDE

18. IF CHRONIC TOTAL OCCLUSION=YES AND ANGINA?= YES AND MILD CHEST PAIN ? = YES

THEN TREATMENT =

ADOPT HEALTHY LIFESTYLE

REGULAR CHECKUPS

ASPIRIN

ATENOLOL

ATORVASTATIN

NORVASC

NITRATES

19. IF CORONARY MICROVASCULAR DISEASE=YES AND SUFFERING FROM PAIN
? = YES AND LIFESTYLE CHANGES DIDN'T WORK ? = YES

THEN TREATMENT =

CHOLESTEROL MEDICATION

BP MEDICATION

NITROGLYCERIN

NORVASC

ATENOLOL

ANTIPLATELET MEDICINE

20. IF CARDIAC TAMPONADE=YES AND CHEST X-RAY SHOWS ENLARGED
HEART ? = YES AND DECREASE IN BP WHILE INHALING ? = YES AND
ABNORMALITIES IN 2D ECHO ? = YES

THEN TREATMENT =

REQUIRED HOSPITALIZATION

BED-REST WITH LEG ELEVATION

DOBUTAMINE

IV FLUIDS

EXTERNAL OXYGEN SUPPLY

PERICARDIOCENTESIS

21. IF PULMONARY EMBOLISM=YES AND BLOOD OXYGEN IS LOW ? = YES AND
D-DIMER BLOOD RESULT IS HIGH? = YES AND CONSUME ALCOHOL ?=YES

THEN TREATMENT =

STOP CONSUMING ALCOHOL

HEPARIN

WARFARIN

REGULAR CHECKUPS

22. IF CARDIOGENIC SHOCK=YES AND DECREASE IN BP WHILE INHALING ? = YES AND ABNORMALITIES IN ECG ? = YES AND X-RAY SHOWS FLUID IN LUNGS = YES

THEN TREATMENT =

EMERGENCY LIFE SUPPORT NEEDED

IV FLUIDS

ASPIRIN

VASOPRESSORS

ANTIPLATELET MEDICATION

BLOOD THINNING MEDICATION

23. IF ATHEROSCLEROSIS=YES AND HIGH BP?= YES AND PHYSICALLY ACTIVE?=YES AND BEING STRESSFUL ? = YES AND TYPE 2 DIABETES?=YES

THEN TREATMENT =

HEALTHY LIFESTYLE

STRESS MANAGEMENT

BE PHYSICALLY ACTIVE

METFORMIN

NITRATES

ATENOLOL

NORVASC

ANTI-CLOTTING MEDICINE

24. IF HEART VALVE STENOSIS=YES AND HIGH FEVER ? = YES AND IRREGULAR HEART BEAT ? = YES

THEN TREATMENT =

REGULAR CHECKUPS WITH DOCTOR

DIURETICS

ANTICOAGULANTS

ANTIBIOTICS

ATENOLOL

NORVASC

25. IF EISENMENGER SYNDROME=YES AND BLOOD OXYGEN IS LOW ?=YES
AND CHEST X-RAY SHOWS ENLARGED HEART ? =YES AND ABNORMALITIES IN
THE COMPLETE BLOOD COUNT?=YES

THEN TREATMENT =

ANTIBIOTICS

ANTICOAGULANTS

DIURETICS

DUAL ENDOTHELIN RECEPTOR ANTAGONISTS

IRON SUPPLEMENTS

SUPPLEMENTAL OXYGEN

26. IF MITRAL VALVE PROLAPSE=YES AND HIGH BP?= YES AND CHEST X-RAY
SHOWS ENLARGED HEART ? = YES AND ABLE TO DO STRESS TESTS ? = YES

THEN TREATMENT =

ATENOLOL

DIURETICS

AMIODARANE

ANTI-COAGULANTS

27. IF RHEUMATIC HEART DISEASE=YES

THEN TREATMENT =

PENCILLIN

ANTIBIOTICS

ASPIRIN

STEROIDS

28. IF VENOUS THROMBOEMBOLISM=YES AND AGE? >65 AND OBESE ? = YES
AND LOW OXYGEN LEVELS USING PULSE OXIMETRY ? = YES

THEN TREATMENT =

APIXABAN

RIVAROXABAN

29. IF HYPERTROPHIC OBSTRUCTIVE CARDIO MYOPATHY=YES

THEN TREATMENT =

ATENOLOL

DYSOPYRAMIDE

NORVASC

30. IF MYOCARDIAL INFARCTION=YES AND LITTLE OR NO BLOOD FLOW IN SOME AREAS=YES AND INCREASED CLOT FORMATION IN BLOOD=YES

THEN TREATMENT =

SUPPLEMENTARY OXYGEN

ANTI-CLOTTING MEDICATIONS

NITROGLYCERIN

THROMBOLYTIC (CLOT-BUSTING) MEDICATIONS

ANTI-ARRHYTHMIA MEDICATIONS

PAIN MEDICATIONS

6.PROGRAM IMPLEMENTATION

The program has been implemented from scratch using C++. Project1-A05244809.cpp contains the source code. Attacks_BW() function in this program implements the backward chaining algorithm and passes the conclusion (name of the diagnosed disease) to the Prevention_FW(string disease). Prevention_FW then proceeds to process the passed variable to recommend relevant medications. The sections, 6.1 and 6.2 below explain how each function works in detail

6.1 Backward Chaining

Attacks_BW() function first calls initializeDS() function followed by Process() function. initializeDS() function initializes all the data structures required to perform backward chaining and Process() function implements backward chaining.

6.1.1 initializeDS()

The data structures used in backward chaining algorithm, initialized, or used in this function are:

Variable name	Data structure	Datatype	Used for
rule_list	Vector	struct Rule_type{ int ruleIndex; vector<string>conditions; string conclusion;} 	Storing rules for diagnosing diseases.
cncl_var_list	Vector	struct Conclusion_list { int ruleno; string varname; }; 	Storing the conclusion that each rule number can produce.
varhashMap	unordered_map	<string, string>	Used for tracking if variables have been initiated or not. With each update in a clause in a premise

			the varhashMap is also updated.
clausevarlist	Map	<int,string>	Stores all variable's values corresponding to a rule. All these variables in clause variable list need to be analysed to see if a rule holds.
derivedGlobalVariableList	unordered_map	<string,string>	Used in cases where sub rules need to be evaluated to give a conclusion. Used during recursive calls, that is when an if clause is a conclusion of another rule.

6.1.2 Process()

Process function processes the “goal” to return a conclusion/diagnosis .The goal in our case is disease. For each conclusion in the cncl_var_list the following functions will execute:

- search_con(string variable, int index): with passing the goal variable as a parameter (variable). It will find the matching variable in the conclusion list and the corresponding rule number, Ri.
- rule_to_clause (int ruleno) with passing rule number as a parameter. This function will convert the rule number, Ri, to clause number, Ci. Here we have used the formula:

$$Ci = 15 * ((Ri / 10) - 1) + 1$$

We have assigned 15 slots for each rule in the Clause Variable list. Since the upper limit of variables in clause variable list was 12, we have kept a buffer of 3 more as a backup. Our rules numbered as 10,20...etc hence we have ruleno/10 in our formula.

- update_VL (integer clauseno): It will all questions and will instantiate all (maximum of 15) variables starting from the location Ci in the variable list. If the variable is not in the variable list, it could be in the then clause of some rule (we use cncl_var_list to check this here. It stores all conclusions as a list). If found in, then side of rule we call Process (variable) to find that variable's value-it will be a recursive call.

- `validate_Ri(int ruleNo, string& conclusion)`: with passing, `ruleNo`, to it. It will check if the values of the variables in the 'if' clauses of the rule, `ruleNo`, are satisfied with the values in the variable list and the derived global variable list. If they do, it will assign conclusion of the rule to the variable conclusion; otherwise, it will not assign any value and then it will return.

6.2 Forward Chaining

It receives the disease(output of backward chaining as an input). And proceeds to call `fw_initializeDS()` and `fw_Process(dis)`

6.2.1 `fw_initializeDS()`

Initializes or used the following globally declared variables:

Variable name	Data structure	Datatype	Used for
<code>fw_rule_list</code>	Vector	struct forward_rule_type {int ruleIndex; vector<string> conditions; vector<string> treatment;};	Storing rules which produce set of treatments corresponding to health conditions.
<code>fw_varhashMap</code>	unordered_map	<string, string>	Used for tracking if variables have been initiated or not. With each update in a clause in a premise the varhashMap is also updated.
<code>fw_clausevarlist</code>	Map	<int,string>	Stores all variable's values corresponding to a rule. All these variables in clause variable list need to be analysed to see if a rule holds.
<code>derived_con_list</code>	vector	string	When we get a rule that satisfies all conditions we store the treatment methods in this vector.

6.2.2 fw_Process()

- search_cvl(string variable): this function will search for an entry in the clause variable list and find the entry which matches the argument variable and returns the clause number, Ci, corresponding to the matching entry.
After finding the variable in clause variable list fw_update_VL(int clauseno) and clause_to_rule(int clauseno) is called.
- fw_update_VL(int clauseno): For each variable it will check if the variable is instantiated in the fw_varhashMap or not. If not, it will ask the user to provide the values of variables and instantiate them.
- clause_to_rule(int clauseno): Clauseno passed here is converted to rule number using formula:
$$\text{ruleno} = ((\text{clauseno}/10) + 1) * 10;$$

Since rule numbers are sequenced like 10,20,30,40,50, we multiply by 10. Clause variable list 10 slots have been assigned.

- validate_Ri(int ruleno): Evaluates each variable in the rule with fw_varhashMap and stores derived treatments in derived_con_list and in the queue if the rules are evaluated to be true.

7.SOURCE CODE

```
//Have used Option 1, The algorithm

#include <iostream>
#include <vector>
#include <string>
#include <list>
#include <map>
#include <unordered_map>
#include <stack>
#include <algorithm>
#include <string.h>
#include <queue>
#include <string>

#include <chrono>
using namespace std;
using namespace std::chrono;

// Defining a structure to represent a rule
struct Rule_type {
    int ruleIndex; // Index of the rule
    vector<string> conditions;
    string conclusion;
};
```

```

// Define a structure to represent a conclusion
struct Conclusion_list {
    int ruleno;
    string varname;
};

struct stack_type {
    int ruleno;
    int clauseno;
};

struct Derived_global{
    string variable;
    string instantiated;
    string value;
};

//forward chaining structures

using namespace std;
struct forward_rule_type {
    int ruleIndex; // Index of the rule
    vector<string> conditions;
    vector<string> treatment;
};

// Initialize the Backward chaining rule list
vector<Rule_type> rule_list = {
{10,    {"ARRHYTHMIA=YES" , "SWELLING=YES" , "WEIGHTGAIN=YES"},
"ISCHEMIC CARDIOMYOPATHY"},

{20,    {"CHESTPAIN=YES", "FATIGUE=YES" , "SHORTNESSOFBREATH=WLD" , "FEVER=NO"
, "BLRYVISION=YES", "HEARTPALPITATION=YES"}},
"ARRHYTHMIA"},

//change after stack test
//{10, {"CHESTPAIN=YES" , "FATIGUE=NO", "SHORTNESSOFBREATH=YES"
, "FAINTING=YES", "NOISYGASPING=NO", "WEAKENINGINLEFTVENTRICLE=NO",
"NECKTIGHTNESS=YES", "HEARTPALPITATION=YES", "CARDIACARST=YES"}, "VENTRICULAR
TACHYCARDIA"},

//{20, {"CHESTPAIN=YES" , "FATIGUE=NO" ,
"SHORTNESSOFBREATH=YES", "FAINTING=YES", "NOISYGASPING=NO",
"WEAKENINGINLEFTVENTRICLE=YES", "LOWBP=YES", "HEARTPALPITATION=YES"},
"TAKOTSUBO CARDIOMYOPATHY (BROKEN HEART SYNDROME)"},

```

```

{30,    {"CHESTPAIN=YES", "FATIGUE=NO", "SHORTNESSOFBREATH=YES" ,
"FAINTING=YES", "NOISYGASPING=YES", "SEIZURES=YES"}, "LONG QT SYNDROME"},

{40,    {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ALLTIME",
"FEVER=NO", "SHOULDERPAIN=NO", "BACKPAIN=NO", "DIZZINESS=YES",
"IRREGULARHEARTBEAT=NO", "NAUSEA=YES"}, "PAROXYSMAL ATRIAL FIBRILLATION"},

{50,    {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=NEVER",
"SWELLING=YES", "BLUESKIN=YES",
"DIZZINESS=YES", "INCREASEDABDOMINALSIZE=YES"},
"PULMONARY HYPERTENSION"},

{60,    {"CHESTPAIN=YES",
"FATIGUE=YES", "SHORTNESSOFBREATH=ALLTIME", "FEVER=NO", "SHOULDERPAIN=NO",
"BACKPAIN=YES", "COUGH=YES", "SCRTHVOICE=YES"}
,"AORTIC ANEURSYM"},

{70,    {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ALLTIME",
"FEVER=NO", "SHOULDERPAIN=NO", "BACKPAIN=NO", "DIZZINESS=YES",
"IRREGULARHEARTBEAT=YES"},
"VALVULAR HEART DISEASE"},

{80,    {"CORONARY ARTERY DISEASE=YES", "NAUSEA=YES", "INDIGESTION=YES",
"FAINTING=YES" ,"HEAVYSWEATING=YES"},
"ACUTE CORONARY SYNDROME"},

{90,    {"CHESTPAIN=YES", "FATIGUE=YES" , "SHORTNESSOFBREATH=ALLTIME " ,
"FEVER=NO" , "SHOULDERPAIN=YES" , "HEARTPALPITATION=YES" , "NECKJWPAIN=YES"},
"CORONARY ARTERY DISEASE"},

{100,   {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ALLTIME",
"FEVER=YES", "HEARTMURMUR=NO", "SWOLLENJOINTS=YES" ,
"UNCONTROLLEDMOVEMENT=YES", "SKINLUMPS=YES"}},
"RHEUMATIC HEART DISEASE"},

{110,   {"PULMONARY EMBOLISM=YES" , "REDLIMB=YES", "TENDERNESSOFTHIGH=YES",
"WARM SKIN=YES"},
"VENOUS THROMBOEMBOLISM"},

{120,   {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ALLTIME",
"FEVER=YES", "HEARTMURMUR=YES", "SKINBUMPS=NO", "BLOODINCOUGH=YES",
"DISCOLOREDSKIN=YES"}},
"PULMONARY EMBOLISM"},

{130,   {"CHESTPAIN=YES" , "FATIGUE=YES" , "SHORTNESSOFBREATH=ALLTIME",
"FEVER=YES", "HEARTMURMUR=YES", "SKINBUMPS=YES" }},
"ENDOCARDITIS"},

```

```

/*{140, {"ARRHYTHMIA=YES" , "SWELLING= YES" , "WEIGHTGAIN=YES"},
"ISCHEMIC CARDIOMYOPATHY"},

{150, {"CHESTPAIN=YES", "FATIGUE=YES" , "SHORTNESSOFBREATH=LYINGDOWN" ,
"FEVER=NO" , "BLRYVISION=YES", "HEARTPALPITATION=YES"},
"ARRHYTHMIA"},*/
{140, {"CHESTPAIN=YES" , "FATIGUE=NO", "SHORTNESSOFBREATH=YES"
,"FAINTING=YES", "NOISYGASPING=NO", "WEAKENINGINLEFTVENTRICLE=NO",
"NECKTIGHTNESS=YES", "HEARTPALPITATION=YES", "CARDIACARST=YES"}, "VENTRICULAR
TACHYCARDIA"},

{150, {"CHESTPAIN=YES" , "FATIGUE=NO" ,
"SHORTNESSOFBREATH=YES", "FAINTING=YES", "NOISYGASPING=NO",
"WEAKENINGINLEFTVENTRICLE=YES", "LOWBP=YES", "HEARTPALPITATION=YES"},
"TAKOTSUBO CARDIOMYOPATHY"},

{160, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=LYINGDOWN",
"FEVER=NO", "BLRYVISION=NO", "DIFFICULTYINWALKING=YES" , "SWELLING=YES" ,
"PREFERSITTING=YES"}},
"HEART VALVE STENOSIS"},

{170, {"CHESTPAIN=YES" , "FATIGUE=YES", "SHORTNESSOFBREATH=LYINGDOWN" ,
"FEVER=NO" , "BLRYVISION=NO" , "DIFFICULTYINWALKING=NO", "FAINTING=YES" ,
"IRREGULARHEARTBEAT=YES" , "HEARTPALPITATION=YES" , "SEIZURES=YES" },
"BRUGADA SYNDROME"},

{180, {"PERICARDITIS=YES", "INFECTIONORWOUND=YES", "CANCER=YES"},
"CARDIAC TAMPONADE"},

{190, {"CHESTPAIN=YES" , "FATIGUE=YES" , "SHORTNESSOFBREATH=LYINGDOWN" ,
"FEVER=YES" , "SWEATCHILLS=YES" , "HEARTPALPITATION=YES", "DIZZINESS=YES"},
"PERIPHERAL ARTERY DISEASE"},

{200, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ACTIVITY",
"FAINTING=NO", "SLURRED SPEECH=NO", "SLEEP PROBLEM=YES", "LACK OF ENERGY=YES"},
"CORONARY MICROVASCULAR DISEASE"},

{210, {"CHESTPAIN=YES", "FATIGUE=YES" , "SHORTNESSOFBREATH=ACTIVITY" ,
"FAINTING=NO" , "SLURRED SPEECH=NO" , "SLEEP PROBLEM=NO" , "DIZZINESS=YES" ,
"IRREGULARHEARTBEAT=YES" , "NAUSEA=YES" , "HEARTPALPITATION=YES" ,
"UPRARM PAIN=YES"}},
"CHRONIC TOTAL OCCLUSION"},

{220, {"CHESTPAIN=YES" , "FATIGUE=YES", "SHORTNESSOFBREATH=ACTIVITY" ,
"FAINTING=NO" , "SLURRED SPEECH=YES" , "ONE EYE VISION LOSS=YES" , "LEG CRMP=YES"},
"ATHEROSCLEROSIS"},

```

```

{230, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ACTIVITY",
"FAINTING=YES", "DIZZINESS=YES", "HEARTMURMUR=YES", "SWELLING=YES",
"NMBNESS=NO"},
"HYPERTROPHIC CARDIOMYOPATHY"},

{240, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ACTIVITY",
"FAINTING=YES", "DIZZINESS=YES", "HEARTMURMUR=YES", "SWELLING=YES",
"NMBNESS=YES", "FAMILYHISTORYOFCARDIACARREST= NO",
"FAMILYHISTORYOFMITRALVALVEPROLAPSE=YES"},
"MITRAL VALVE PROLAPSE"},

{250, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ACTIVITY",
"FAINTING=YES", "DIZZINESS=YES", "HEARTMURMUR=YES", "SWELLING=YES",
"NMBNESS=YES", "FAMILYHISTORYOFCARDIACARREST=YES"},
"HYPERTROPHIC OBSTRUCTIVE CARDIO MYOPATHY"},

{260, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ACTIVITY",
"FAINTING=YES", "DIZZINESS=YES", "HEARTMURMUR=NO", "BLOODINCOUGH=YES",
"BLUECOLORSKIN=YES", "HEARTPALPITATION=YES"},
"EISENMENGER SYNDROME"},

{270, {"MYOCINFCTN=YES", "LESSURIN=YES", "PALESKN=YES"},
"CARDIOGENIC SHOCK"},

{280, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=SUDDEN",
"SWEATCHILLS=YES", "BDYPAIN=YES"}, "MYOCARDIAL INFARCTION"},

{290, {"CHESTPAIN=YES",
"FATIGUE=YES", "SHORTNESSOFBREATH=DURINGEPISODESOFRAPIDHEARTRATE", "FAINTING
=YES", "DIZZINESS=YES", "HEARTPALPITATION=YES", "ANXIETY=YES"}, "WOLFF-
PARKINSON-WHITE SYNDROME"},

{300, {"CHESTPAIN=YES", "FATIGUE=YES", "SHORTNESSOFBREATH=ACTIVITY",
"FAINTING=YES", "DIZZINESS=YES", "HEARTMURMUR=NO", "BLOODINCOUGH=YES",
"SLEEP PROBLEM=YES", "SWELLING=YES"}, "AORTIC STENOSIS"}

};

//forward chaining rules
vector<forward_rule_type> fw_rule_list = {

{10, {"CORONARY ARTERY DISEASE=yes", "High cholesterol?=yes", "Smoking?=yes",
"over weight?=yes"}, {"Stop smoking", "Include healthy diet", "Maintain
healthy lifestyle", "Exercise", "Atorvastatin", "Aspirin", "Vasodilators"}},
{20, {"AORTIC STENOSIS=yes", "Severe symptoms=no", "Pulmonary
congestion=yes", "Angina?=yes"}, {"Healthy lifestyle", "Mild
exercise", "Lisinopril", "Carvedilol", "Furosemide", "Digoxin"}},

```

```

{30, {"ARRHYTHMIA=yes" , "Angina?=yes" , "High BP?=yes"},
{"Verapamil","propranolol","warfarin"}},
{40, {"HYPERTROPHIC CARDIOMYOPATHY=yes" , "High cholesterol?=yes"}, {"Healthy
food and
lifestyle","atenolol","disopramide","verapamil","warfiarin","atorvastatin"}},
{50, {"PERIPHERAL ARTERY DISEASE=yes" , "Smoking?=yes" , "physically
active?=no" , "High cholesterol?=yes" , "type 2 diabetes?=yes" , "have High
BP?=yes"}, {"Start with gentle physical
activity","aspirin","atorvastatin","meglitinides","warfarin","cilostazol"}},
{60, {"AORTIC ANEURYSM=yes" , "High BP?=yes" , "High cholesterol?=yes" },
{"Atenolol","losartan","atorvastatin"}},
{70, {"VALVULAR HEART DISEASE=yes" , "High BP?=yes"},
{"Digidoxin","atenolol","amlodipine","diuretics","lisinopril"}},
{80, {"ENDOCARDITIS=yes" , "used antibiotics for 2-6 weeks ?=yes" , "symptoms
reduced ?=no"}, {"Repair damaged heart valve","aortic valve replacement
surgery","draining abscesses and repairing fistulas"}},
{90, {"PULMONARY HYPERTENSION=yes" , "are you pregnant?=no" , "sleep apnea
?=yes"},
{"Epoprostenol","riociguat","bosentan","sildenafil","amlodipine","warfarin","d
igoxin"}},
{100, {"WOLFF-PARKINSON-WHITE SYNDROME=yes" , "have symptoms often?=yes"},
{"Catherer ablation","cardioversion procedure","adinosine"}},
{110, {"ACUTE CORONARY SYNDROME=yes" , "High cholesterol?=yes" , "High
BP?=yes" , "Arrhythmia?=yes"},
{"Aspirin","nitroglycerin","thrombolytics","metoprolol","lisinopril","irbesart
in","ezetimibe","atorvastatin"}},
{120, {"LONG QT SYNDROME=yes" , "want to be athlete or be more physically
active ?=no"}, {"Healthy lifestyle","nadolol","mexiletine"}},
{130, {"PAROXYSMAL ATRIAL FIBRILLATION=yes" , "Smoking?=yes" , "consume
alcohol ?=yes" , "being stressful ?=yes"}, {"Stop smoking","limit alcohol
consumption","start a healthy lifestyle","practice stress-
relieving","digoxin","procainamide","warfarin"}},
{140, {"TAKOTSUBO CARDIOMYOPATHY=yes" }, {"Restrict tobacco use","manage
stress","aspirin","diuretics","benzapraxil","atenolol"}},
{150, {"VENTRICULAR TACHYCARDIA=yes" , "High BP?=yes" , "High
cholesterol?=yes" , "Smoking?=yes" , "consume alcohol ?=yes"}, {"Eat low fat,
low salt diet","exercise often","stop using
tobacco","norvasc","atenolol","amiodarone"}},
{160, {"BRUGADA SYNDROME=yes" , "Arrhythmia?=yes" , "high fever ?=yes"},
{"Quinidine","fever reducing medicine"}},
{170, {"ISCHEMIC CARDIOMYOPATHY=yes" , "obese ?=yes" , "family history of this
disease ?=yes" , "High BP?=yes" , "High cholesterol?=yes" , "consume drugs
?=yes" , "Smoking?=yes" , "swelling in the legs ?=yes" }, {"Avoid drugs","stop
smoking","eat healthy
diet","atenolol","norvasc","atorvastatin","warfarin","furosemide"}},
{180, {"CHRONIC TOTAL OCCLUSION=yes" , "Angina?=yes" , "mild chest pain
?=yes"}, {"Adopt healthy lifestyle","regular
checkups","aspirin","atenolol","atorvastatin","norvasc","nitrates"}},

```

```

{190, {"CORONARY MICROVASCULAR DISEASE=yes" , "suffering from pain ?=yes" ,
"lifestyle changes didn't work ?=yes"}, {"Cholesterol medication","bp
medication","nitroglycerin","norvasc","atenolol","antiplatelet medicine"}},
{200, {"CARDIAC TAMPONADE=yes" , "chest X-ray shows enlarged heart ?=yes" ,
"decrease in BP while inhaling ?=yes" , "abnormalities in 2D echo ?=yes"},
{"Required hospitalization","bed-rest with leg elevation","dobutamine","iv
fluids","external oxygen supply","pericardiocentesis"}},
{210, {"PULMONARY EMBOLISM=yes" , "blood oxygen is low ?=yes" , "D-dimer blood
result is high?=yes" , "consume alcohol ?=yes"}, {"Stop consuming
alcohol","heparin","warfarin","regular checkups"}},
{220, {"CARDIOGENIC SHOCK=yes" , "decrease in BP while inhaling ?=yes" ,
"abnormalities in ECG ?=yes" , "X-ray shows fluid in lungs=yes"}, {"Emergency
life support needed","iv fluids","aspirin","vasopressors","antiplatelet
medication","blood thinning medication"}},
{230, {"ATHEROSCLEROSIS=yes" , "High BP?=yes" , "physically active?=yes" ,
"being stressful ?=yes" , "type 2 diabetes?=yes"}, {"Healthy
lifestyle","stress management","be physically
active","metformin","nitrates","atenolol","norvasc","anti-clotting
medicine"}},
{240, {"HEART VALVE STENOSIS=yes" , "high fever ?=yes" , "irregular heart beat
?=yes"}, {"Regular checkups with
doctor","diuretics","anticoagulants","antibiotics","atenolol","norvasc"}},
{250, {"EISENMENGER SYNDROME=yes" , "blood oxygen is low ?=yes" , "chest X-ray
shows enlarged heart ?=yes" , "abnormalities in the complete blood
count?=yes"}, {"Antibiotics","Anticoagulants","Diuretics","Dual endothelin
receptor antagonists","Iron supplements","Supplemental oxygen"}},
{260, {"MITRAL VALVE PROLAPSE=yes" , "High BP?=yes" , "chest X-ray shows
enlarged heart ?=yes" , "able to do stress tests ?=yes"},
{"Atenolol","diuretics","amiodarone","anti-coagulants"}},
{270, {"RHEUMATIC HEART DISEASE=yes"},
{"Pencillin","antibiotics","aspirin","steroids"}},
{280, {"VENOUS THROMBOEMBOLISM=yes" , "Age greater than 65=yes" , "obese
?=yes" , "low oxygen levels using pulse oximetry ?=yes" }, {
"apixaban","rivaroxaban"}},
{290, {"HYPERTROPHIC OBSTRUCTIVE CARDIO MYOPATHY=yes"},
{"Atenolol","dysopyramide","norvasc"}},
{300, {"MYOCARDIAL INFARCTION=yes" , "Little or no blood flow in some
areas=yes" , "increased clot formation in blood=yes" }, {"Supplementary
oxygen","Anti-clotting medications","Nitroglycerin","Thrombolytic (clot-
busting) medications","Anti-arrhythmia medications","Pain medications"}}

};

//Backward chaining datastructures
vector<Conclusion_list> cncl_var_list(30);
unordered_map<string, string> varhashMap(30);
map<int,string> clausevarlist[400];
stack<stack_type> conclusionStack;

```

```

vector<string> conclusionList;
unordered_map<string,string> derivedGlobalVariableList(30);
//Forward Chaining datastructures
queue<string> con_var_q;
unordered_map<string, string> fw_varhashMap(30); // variable list
vector<string>derived_con_list{}; //derived_con_list
std::unordered_map<int,int>cls_var_pointer; //pointer
map<int,string> fw_clausevarlist[400];
queue<string>q;

//Backward chaining functions
int Attacks_BW();
string Process(string goal);
void initializeDS();
int search_con(string variable,int index);
int rule_to_clause(int ruleno);
void update_VL(int clauseno);
void validate_Ri(int ruleNo, string& conclusion);
//Forward Chaining functions
int Prevention_FW(string disease);
void fw_Process(string value);
void fw_initializeDS();
void search_cvl(string variable);
void clause_to_rule(int clauseno);
void fw_update_VL(int clauseno);
void validate_Ri(int ruleno);

void initializeDS(){
    //Intializing the conclusionList??
    for(Rule_type rule : rule_list){
        string conclusion = rule.conclusion;
        conclusionList.push_back(conclusion);
    }
    //Intializing the conclusion variable list
    for(int i=0;i<30;i++){
        cncl_var_list[i].ruleno = 10*(i+1);
        cncl_var_list[i].varname = "disease";
    }

    //Initializing the variable list
    for(Rule_type rule : rule_list){
        vector<string> condList = rule.conditions;
        for(string cond : condList){
            size_t pos = cond.find("=");

```



```

        string cond_name = cond.substr(0,pos);
        string cond_val = cond.substr(pos+1);
        //if cond_name not there in conclusion list add it to variable
list
        auto it = find(conclusionList.begin(), conclusionList.end(),
cond_name);
        if (it == conclusionList.end()) {
            varhashMap[cond_name] = "NI";
        }
        else {
            std::size_t index = std::distance(conclusionList.begin(), it);
            cncl_var_list[index].varname = cond_name;
        }
    }
}

int clausenum=1;
//Initializing Clause Variable list
for(Rule_type rule : rule_list){
    vector<string> condList = rule.conditions;
    array<string,15> temparr;
    for(int i=0;i<15;i++){
        temparr[i]="";
    }
    int ind=0;
    for(string cond : condList){
        size_t pos = cond.find("=");
        string cond_name = cond.substr(0,pos);
        string cond_val = cond.substr(pos+1);
        temparr[ind] = cond_name;
        ind++;
    }
    for(int i=0;i<15;i++){
        clausevarlist->insert(pair<int,string>(clausenum+i,temparr[i]));
    }
    clausenum += 15;
}

/*cout << "print variable list" << "\n";
for (auto i = varhashMap.begin(); i != varhashMap.end(); i++)
    cout << i->first << " \t\t\t" << i->second << endl;

cout << "printing clause variable list" << "\n";
for (auto i = clausevarlist->begin(); i != clausevarlist->end(); i++)
    cout << i->first << " \t\t\t" << i->second << endl;

cout << "printing conclusion variable list" << "\n";
for(Conclusion_list c:cncl_var_list) {

```

```

        cout<<"\n Rule: \n"<<c.ruleno;
        cout<<"\n conc variable: \n"<<c.varname;
    }*/
}

// finds the given variable name in conclusion list
int search_con(string variable,int index){
    /* for(int i=0; i < cncl_var_list.size(); i++){
        int res  = (cncl_var_list[index].varname).compare(variable);
        if(res == 0){
            return cncl_var_list[index].ruleno;
        }
    }
    return -1; */

    int res  = (cncl_var_list[index].varname).compare(variable);
    if(res==0){
        return cncl_var_list[index].ruleno;
    }
    else{
        return -1;
    }
}

// finds the clause no for given rule no
int rule_to_clause(int ruleno){
    stack_type item;
    item.ruleno = ruleno;
    item.clauseno = 15 * ((ruleno / 10) - 1) + 1;

    // Push the item onto the conclusionStack
    conclusionStack.push(item);

    return item.clauseno;
}

void update_VL(int clauseno){

    string userinput;

    //for each variable in clausevarlist we have to find its value in
    varhashMap and cncl_var_list
    for(int clnum=clauseno;clnum<clauseno+15;clnum++){

```

```

        auto questionvar = clausevarlist->find(clnum);
        string questoask = questionvar->second; // //for example cp in clause
variable list
        if(!(questoask == "")){

            bool inconclusionvarList=false;
            for(int i=0;i<30;i++){
                if(cncl_var_list[i].varname == questoask){
                    inconclusionvarList=true;
                    break;
                }
            }

            if(inconclusionvarList){

                Process(questoask);

            }
            else{
                auto it = varhashMap.find(questoask);
                string val = it->second;
                int res = val.compare("NI");
                if(res == 0){

                    cout << "\nDo you have " << questoask << " ? " << "\n";
                    if(questoask=="SHORTNESSOFBREATH") {
                        cout << "\nPlease enter your symptom from anyone
below:\nWLD : If you have symptom while lying down\n";
                        cout << "\nNEVER: If you never have this symptom \n";
                        cout << "\nYES : If you are unsure when you are short of
breath \n";

                        cout << "\nALLTIME : If you are experiencing shortness of
breath all the time \n";
                        cout << "\nLYINGDOWN : If you experience shortness of
breath while lying down occasionally \n";
                        cout << "\nACTIVITY : If you experience shortness of
breath during physical activity \n";
                        cout << "\nSUDDEN : If you experience shortness of breath
suddenly \n";

                        cout << "\nDURINGEPISODESOFRAPIDHEARTRATE : If you
experience shortness of breath whenever heart rate increases\n";
                    }
                    else{
                        cout<< "\nEnter YES or NO as answer \n";
                    }
                    cin >> userInput;
                    varhashMap[questoask] = userInput;

                }
            }
        }
    }

```

```

    }
}
else{
    break;
}
}
}

void validate_Ri(int ruleNo, string& conclusion) {
    bool conditionSatisfied;
    bool allconditionsSatisfied=true;
    Rule_type checkedRule;

    for (const Rule_type& rule : rule_list) {
        if (rule.ruleIndex == ruleNo) {
            checkedRule=rule;
            break;}
    }

    for(string cond : checkedRule.conditions){
        conditionSatisfied=true;
        size_t pos = cond.find("=");
        string cond_name = cond.substr(0,pos);
        string cond_val = cond.substr(pos+1);

        if (varhashMap[cond_name].compare(cond_val)!=0 &&
derivedGlobalVariableList[cond_name].compare(cond_val)!=0) { //check case
lower,upper
            conditionSatisfied = false;
            break;
        }

    }
    if(!conditionSatisfied){
        allconditionsSatisfied=false;
    }

    if(allconditionsSatisfied){

        conclusion=checkedRule.conclusion;
        derivedGlobalVariableList[conclusion]="YES";
        stack_type top;
        top=conclusionStack.top();
        if(top.ruleno==checkedRule.ruleIndex){
            conclusionStack.pop();
        }
    }
}

```

```

        //cout<< "all conditions satisfied: " <<conclusion;
    }
    //cout<<"after setting derived
global"<<derivedGlobalVariableList[conclusion];
}

string Process(string goal) {
    //loop through each conclusion variable to find value for goal which is
disease initially
    int i=0;
    string conclusion="";

    while(i<cncl_var_list.size()){

        int rulenum = search_con(goal,i);

        if(rulenum== -1){
            i=i+1;
            continue;
        }

        int clauseno = rule_to_clause(rulenum);

        update_VL(clauseno);

        /*cout << "print variable list" << "\n";
        for (auto i = varhashMap.begin(); i != varhashMap.end(); i++)
            cout << i->first << " \t\t\t" << i->second << endl;*/

        conclusion="";
        validate_Ri(rulenum,conclusion);

        if (!conclusion.empty()) {
            // A conclusion was found, you can save it or use it as
needed
            break; // End the program after updating the Variable
List
        }
        else{
            i=i+1;
        }
    }
    return conclusion;
}

```

```

int Attacks_BW(){
    vector<int> values(10000);
    auto start = high_resolution_clock::now();

    initializeDS();
    cout<< "Welcome to Intelligent Cardiac Diagnosis System \n";
    cout<< "Please answer the following questions so we can diagnose your
disease \n";
    string conclusion = Process("disease");
    cout<< "You have been diagnosed with: \n"<<conclusion;

    // Call the function, here sort()
    sort(values.begin(), values.end());

    // Get ending timepoint
    auto stop = high_resolution_clock::now();

    // Get duration. Substart timepoints to
    // get duration. To cast it to proper unit
    // use duration cast method
    auto duration = duration_cast<microseconds>(stop - start);
    cout<<"\n Time taken by Backward chaining:\n"<<duration.count()
<<"microsec";

    start = high_resolution_clock::now();

    Prevention_FW(conclusion);
    // Call the function, here sort()
    sort(values.begin(), values.end());

    // Get ending timepoint
    stop = high_resolution_clock::now();

    // Get duration. Substart timepoints to
    // get duration. To cast it to proper unit
    // use duration cast method
    duration = duration_cast<microseconds>(stop - start);
    cout<<"\n Time taken by Forward chaining:\n"<<duration.count()
<<"microsec";
    return 0;
}

//forward chaining
void fw_initializeDS(){

    //Initializing the variable list
    for(forward_rule_type rule : fw_rule_list){

```

```

        vector<string> condList = rule.conditions;
        for(string cond : condList){
            size_t pos = cond.find("=");
            string cond_name = cond.substr(0,pos);
            string cond_val = cond.substr(pos+1);

            fw_varhashMap[cond_name] = "NI";
        }
    }

    int clausenum=1;
    //Initializing Clause Variable list
    for(forward_rule_type rule : fw_rule_list){
        vector<string> condList = rule.conditions;
        array<string,10> temparr;
        for(int i=0;i<10;i++){
            temparr[i]="";
        }
        int ind=0;
        for(string cond : condList){
            size_t pos = cond.find("=");
            string cond_name = cond.substr(0,pos);
            string cond_val = cond.substr(pos+1);
            temparr[ind] = cond_name;
            ind++;
        }
        for(int i=0;i<10;i++){
            fw_clausevarlist->insert(pair<int,string>(clausenum+i,temparr[i]));
        }
        clausenum += 10;
    }

    /*cout << "print variable list" << "\n" ;
    for (auto i = fw_varhashMap.begin(); i != fw_varhashMap.end(); i++)
        cout << i->first << " \t\t\t" << i->second << endl;

    cout << "printing clause variable list" << "\n";
    for (auto i = fw_clausevarlist->begin(); i != fw_clausevarlist->end();
i++)
        cout << i->first << " \t\t\t" << i->second << endl;
*/
}

void search_cv1(string variable){
    //cout << "search_cv1 variable name " << " " << variable <<
endl;//ruleno=-1 case??
    for(int i= 0;i<400;i++)

```

```

    {
        for(auto it=fw_clausevarlist[i].begin();
it!=fw_clausevarlist[i].end(); ++it)
        {
            if(it->second == variable)
            {
                int clauseno = it->first;
                //cout << "clauseno" << " " <<clauseno << endl;
                fw_update_VL(clauseno);
                clause_to_rule(clauseno);
                break;
            }
        }
    }
}

void fw_update_VL(int clauseno)
{
    string userinput;
    for(int clnum=clauseno;clnum<clauseno+10;clnum++)
    {

        auto questionvar = fw_clausevarlist->find(clnum);

        string questoask = questionvar->second; // DIS HICOL

        if(!(questoask == "")){

            //cout<< "qn from clause varlist:"<<questoask << "\n";//handle
questoask=="here
            auto it = fw_varhashMap.find(questoask);
            string val = it->second;
            int res = val.compare("NI");//result if variable Instantiated
in variable list or not
            if(res ==0)
            {
                cout<<questoask <<" "<<endl;
                cin>>userinput;
                fw_varhashMap[questoask] = userinput;

                //cout << "value intialized in variable list";

            }

        }
    }
}
else

```



```

        {
            break;
        }
    }

}

void clause_to_rule(int clauseno){

    int ruleno= ((clauseno/10)+1)*10;
    //cout << "rule no " << " " << ruleno << endl;//ruleno=-1 case??
    validate_Ri(ruleno);

}

void validate_Ri(int ruleno)
{

    bool conditionSatisfied;
    bool allconditionsSatisfied=true;
    forward_rule_type checkedRule;

    for (const forward_rule_type& rule : fw_rule_list) {
        if (rule.ruleIndex == ruleno) {
            checkedRule=rule;
            break;
        }
    }

    //cout<< "Rule's validity checking : "<<checkedRule.ruleIndex;

    for(string cond : checkedRule.conditions){
        conditionSatisfied=true;
        size_t pos = cond.find("=");
        string cond_name = cond.substr(0,pos);
        string cond_val = cond.substr(pos+1);

        //cout<<"cond_val "<<cond_val<<endl;
        std::string VLvalue = fw_varhashMap[cond_name];
        //cout<<"VLvalue "<<VLvalue<<endl;
        std::transform(VLvalue.begin(), VLvalue.end(), VLvalue.begin(),
::toupper);
        //cout<<"UPPER VLvalue "<<VLvalue<<endl;
        std::string cond_val_uppr = cond_val;
        //cout<<"cond_val "<<cond_val<<endl;
        std::transform(cond_val_uppr.begin(), cond_val_uppr.end(),
cond_val_uppr.begin(), ::toupper);

```

```

//cout<<"UPPER cond_val_uppr "<<cond_val_uppr<<endl;

    if (VLvalue!=cond_val_uppr) { //check case lower,upper
        conditionSatisfied = false;
        //cout<< "breaks at : "<< cond_name;
        break;
    }
}
if(!conditionSatisfied){
    allconditionsSatisfied=false;
    cout<<"Exercise regularly"<<endl;
}
//cout<< "all conditions satisfied: " <<allconditionsSatisfied;

if(allconditionsSatisfied){
    string con_var="Treatment";
    // cout<< "all conditions satisfied: " <<endl;
    vector<string> trtmntList=checkedRule.treatment;
    cout<< "\n Treatment we recommend is: \n " <<endl;
    for(string trtmnt : trtmntList)
    {

        cout<<trtmnt <<endl;
        derived_con_list.push_back(trtmnt);

    }
    q.push(con_var);
}
}

void fw_Process(string value) {
    for (auto i = fw_varhashMap.begin(); i != fw_varhashMap.end(); i++)
    {
        //cout <<"variable and value ----"<< value <<" " << i->first<< endl;

        if(value == i->first)
        {
            fw_varhashMap[value] = "YES";
            //cout <<"value of var list after update"<< fw_varhashMap[value]
<< endl;

        }

        //cout<<"After update :"<<endl;
        //cout << i->first << " \t\t\t" << i->second << endl;
    }
}

```

```

    }

    search_cvl(value);
}

int Prevention_FW(string dis){





    fw_initializeDS();
    //string variable = "DIS";
    //string dis = "Acute Coronary Syndrome"; // get value from BW
    cout<<"\n";
    cout<<"\nPlease answer few more questions below to get proper treatment:
\n";
    fw_Process(dis);// val of dis
    return 0;
}


int main(){
    Attacks_BW();
    return 0;
}

```

8.COPY OF PROGRAM RUNS

The output obtained from the Ubuntu terminal for 5 testcases has been attached:

Disease	Output
Ischemic Cardiomyopathy:	 Ischemic Cardiomyopathy.txt
Long QT Syndrome:	 Long QT Syndrome.txt
Paroxysmal Atrial Fibrillation:	 Paroxysmal atrial fibrilaltion.txt
Pulmonary Hypertension:	 Pulmonary Hypertension.txt

Aortic Aneurysm:	 Aortic aneurysm.txt
------------------	--

Please open the attachments under output column. These outputs have also been attached separately in case they are not accessible here.

9.ANALYSIS OF THE PROGRAM

9.1 Optimization using hash Maps

We have used hash maps for varHashMap, derivedGlobalVariableList, clausevarlist. Accessing elements using these structures have $O(1)$ time complexity. This reduces time taken from n^3 to n^2 in many places in the program.

9.2 Shortness of breath variable – categorical split

We have added a categorical split in shortness of breath. This variable splits up based on a particular kind of shortness of breath. In our program we prompt the user to enter this very specific detail which will increase the accuracy of the CVD diagnosis:

Please enter your symptom from anyone below:

WLD : If you have symptom while lying down

NEVER: If you never have this symptom

YES : If you are unsure when you are short of breath

ALLTIME : If you are experiencing shortness of breath all the time

LYINGDOWN : If you experience shortness of breath while lying down occasionally

ACTIVITY : If you experience shortness of breath during physical activity

SUDDEN : If you experience shortness of breath suddenly

DURINGEPISODESOFRAPIDHEARTRATE : If you experience shortness of breath whenever heart rate increases

Computational overhead and potential information loss is prevented by using this technique. Taking a decision is complex for CVD diagnosis because all diseases have lot of symptoms in common. But the nature of the symptom varies, and this helps us to bring in the difference between each disease. Taking these small differences enhances the accuracy of the diagnosis.

10.ANALYSIS OF THE RESULTS

10.1 Time Complexity

- Best case time complexity tested with the rule which comes first and has least depth with respect to the decision tree:
 - Time taken for Backward Chaining: 10 -6 sec
 - Time taken for Forward Chaining: approximate 19sec
- Worst case time complexity tested with the rule which comes last and has most depth with respect to the decision tree
 - Time taken for Backward Chaining: 57 sec
 - Time taken for Forward Chaining: 19 sec

10.2 Space Complexity

For calculating space complexity, we introduced a code snippet shown below:

```
{
    vm_usage      = 0.0;
    resident_set = 0.0;

    // the two fields we want
    unsigned long vsize;
    long rss;
    {
        std::string ignore;
        std::ifstream ifs("/proc/self/stat", std::ios_base::in);
        ifs >> ignore >> ignore >> ignore >> ignore >> ignore >> ignore >> ignore >> ignore >> ignore
            >> ignore >> ignore >> ignore >> ignore >> ignore >> ignore >> ignore >> ignore
            >> ignore >> ignore >> vsize >> rss;
    }

    long page_size_kb = sysconf(_SC_PAGE_SIZE) / 1024; // in case x86-64 is configured to use 2MB pages
    vm_usage = vsize / 1024.0;
    resident_set = rss * page_size_kb;
}

int main()
{
    using std::cout;
    using std::endl;

    double vm, rss;
    process_mem_usage(vm, rss);
    cout << "VM: " << vm << " "; RSS: " << rss << endl;
}
```

- We have two primary methods of measuring how much memory is consumed by a process:
- Resident Set Size (RSS): This is a measure of how much memory a process is consuming in our physical RAM, to load all its pages after its execution.
- Virtual Memory Size (VM): This is a measure of how much memory a process can access after its execution. This includes swapped memory, the memory from external libraries, and allocated memory that's not used.
- Our program produced results: VM: 6472 Mb; RSS: 3364 Mb. On an average this is the space our program utilizes. This code was later removed from program as it was intended to measure memory.

11.CONCLUSION

- This project has immense potential to change the medical diagnosis and treatment. We have refined the information collected from the internet about CVDs to make the diagnosis as accurate as possible. Hence the output has maximum accuracy. However, this application must

be tested multiple times with the help of expert doctors, health professionals and patients before being released.

- Furthermore, the symptoms taken into consideration by the system are observed by the user which can be inaccurate. Medical diagnosis is very complex and multiple factors can affect the nature of symptoms user has. Hence this diagnosis is not sufficient to confirm the diagnosis. Follow up tests , including ECGs (Electrocardiograph) ,blood tests, stress tests, Echocardiogram, Chest X-Rays, MRIs etc. are a must to confirm if the patient really has the CVD diagnosed by our system.
- I have learned two powerful methodologies :forward chaining and backward chaining. The significance of forming a decision tree, which can in turn affect the efficiency and accuracy of the application. I have learned that data collection, refinement plays a pivotal role for an expert system development. I also learned the importance of selecting the most apt data structures and the impact it can have on the program efficiency.
- The ability for expert systems to mimic a human brain's decision-making makes it a powerful tool and it can be used across different working sectors. And the most challenging would be to make it available to the world of medicine, as subjects like diagnosis is complex and requires much attention to minor, yet crucial details.

12.REFERENCES

- [1] WHO, "Cardiovascular diseases (CVDs)," [Online]. Available: [https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)).
- [2] cdc, "Centers for Disease Control and Prevention," 9 Jan 2024. [Online]. Available: <https://www.cdc.gov/heartdisease>.
- [3] NIH, "National Heart, Lung, and Blood Institute," NIH, 30 Jun 2023. [Online]. Available: <https://www.nhlbi.nih.gov/health>.
- [4] Penn Medicine, "penn medicine," 11 Jan 2024. [Online]. Available: <https://www.pennmedicine.org/for-patients-and-visitors/patient-information/conditions-treated-a-to-z>.
- [5] Mayo Clinic, 2024. [Online]. Available: <https://www.mayoclinic.org/diseases-conditions/>.
- [6] Cleveland Clinic, "Cleveland Clinic," 2024. [Online]. Available: <https://my.clevelandclinic.org/health/diseases/>.
- [7] Web MD, 2024. [Online]. Available: <https://www.webmd.com/heart-disease/default.htm>.

13.CONTRIBUTION

Gopika Mahadevan

- Gathered information(including symptoms and treatments) for 10 cardiovascular diseases.
- Built 10 branches of the trees required for the program.
- Constructed 10 rules corresponding to 10 diseases.

- Actively participated and contributed to the meetings for integration of rules and trees.
- Implemented 4 functions in the code and worked together with the group for integration.
- Worked to debug code identified and resolved issues. And contributed towards analysing program efficiency.

Sayali Pathak

- Gathered information(including symptoms and treatments) for 10 cardiovascular diseases.
- Built 10 branches of the trees required for the program.
- Constructed 10 rules corresponding to 10 diseases.
- Actively participated and contributed to the meetings for integration of rules and trees.
- Implemented 4 functions in the code and worked together with the group for integration.
- Worked to debug code identified and resolved issues. And contributed towards analysing program efficiency.

Sri Sudha Kambhampati

- Gathered information(including symptoms and treatments) for 10 cardiovascular diseases.
- Built 10 branches of the trees required for the program.
- Constructed 10 rules corresponding to 10 diseases.
- Actively participated and contributed to the meetings for integration of rules and trees.
- Implemented 4 functions in the code and worked together with the group for integration.
- Worked to debug code identified and resolved issues. And contributed towards analysing program efficiency.