<u>The Problem Description:</u>
I have developed an expert system that is capable of assisting a medical professional diagnose a patient with an explicit heart condition. The system will then, based upon that heart condition and its specifications, prescribe the proper treatment for the diagnosis.

<u>The Domain:</u>
The application and development of this system span two separate domains. The application of this system solves a problem in the medical domain. This system will allow medical professionals to spend less time memorizing diseases and their corresponding symptoms and more time solving increasingly abstract medical issues such as personalized patient interaction/care. In order to develop this system and solve the problem in the domain of engineering, I conducted research regarding what heart diseases were common and the symptoms associated with each documented disease. I then found similarities between them in order to construct an efficient decision tree. Once this was done, I was able to start making another decision tree based on the diseases and chained them together so that the proper treatment is linked to its corresponding disease. From here, I developed the software which effectively proved to be my primary solution to the problem in the engineering domain.

<u>Methodologies:</u>
This expert system is comprised of two primary algorithms that control the data-flow of the program. The first algorithm that is compiled when the program is ran is called the backward chaining algorithm. The steps taken to implement the backward chaining structure are as follows:
1. Identify the conclusion
2. Search the conclusion list for the first instance of the conclusion's name. If found, place the rule on the conclusion stack using the rule number and a (1) to represent the clause number. If not found, notify the user that an answer cannot be found.
3. Instantiate the IF clause (i.e., each condition variable) of the statement.
4. If one of the IF clause variables is not instantiated, as indicated by the variable list, and is not a conclusion variable, that is, not on the conclusion list, ask the user to enter a value.
5. If one of the clauses is a conclusion variable, place the conclusion variable's rule number on the top of the stack and go back to step 3.
6. If the statement on top of the stack cannot be instantiated using the present IF-THEN statement, remove the unit from the top of the stack and search the conclusion list for another instance of that conclusion variable's name.408/23/2021
7. If such a statement is found, go back to step 3.
8. If there are no more conclusions left on the conclusion stack with that name, the rule for the previous conclusion is false. If there is no previous conclusion, then notify the user that an answer cannot be found. If there is a previous conclusion, go back to step 6.

9. If the rule on top of the stack can be instantiated, remove it from the stack. If another conclusion variable is underneath, increment the clause number, and for the remaining clauses go back to step 3. If no other conclusion variable is underneath, the system has answered the question. The user can come to a conclusion.

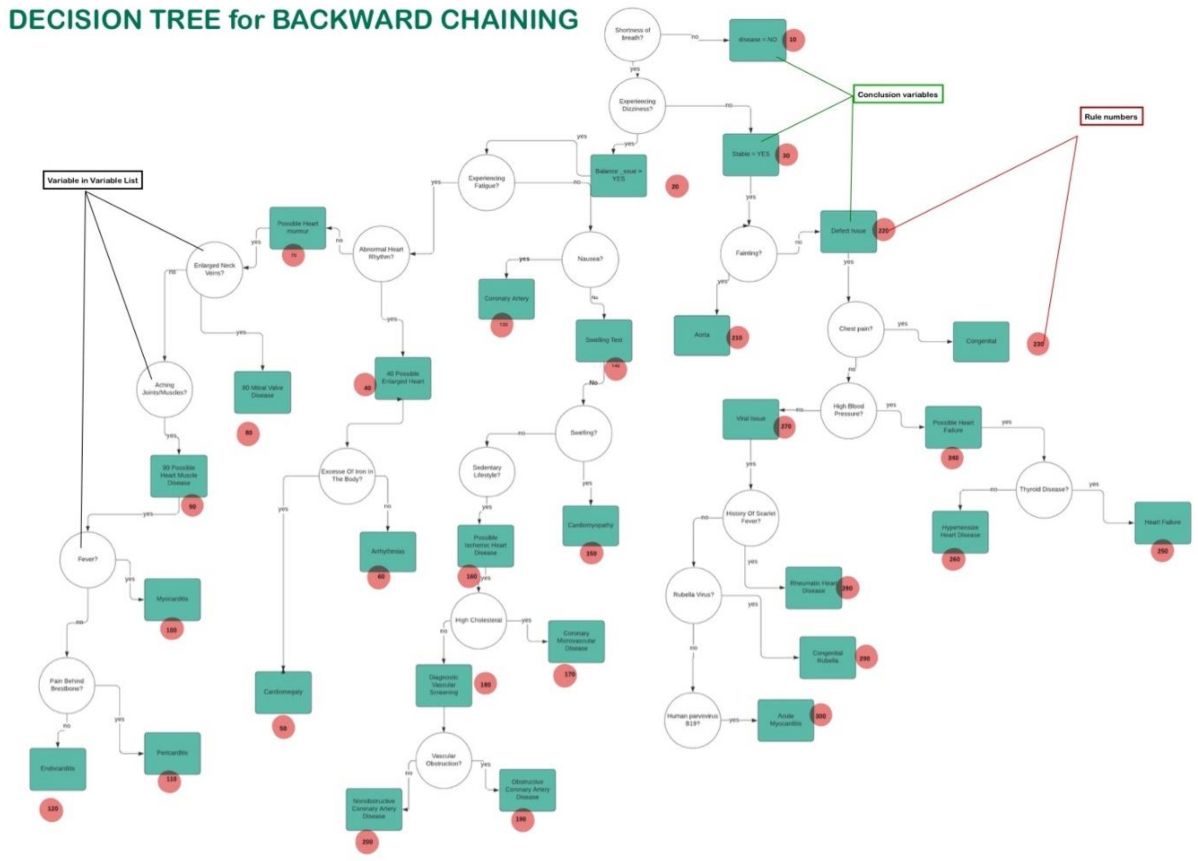
The second algorithm that is compiled when the program is ran is called the forward chaining algorithm. The steps taken to implement the forward chaining structure are as follows:

1. The condition is identified.
2. The condition variable is placed on the conclusion variable queue and its value is marked on the variable list.
3. The clause variable list is searched for the variable whose name is the same as the one in the front of the queue. If found, the rule number and a 1 are placed into the clause variable pointer. If not found, go to step 6.
4. Each variable in the IF clause of the rule that is not already instantiated is now instantiated. The variables are in the clause variable list. If all the clauses are true, the THEN part is invoked.
5. The instantiated THEN part of the variable is placed in the back of the conclusion variable queue.
6. When there are no more IF statements containing the variable that is at the front of the conclusion variable queue, that variable is removed.
7. If there are no more variables on the conclusion variable queue, end the session. If there are more variables, go to step 3. We can use the above concepts to design, in C, a simple forward chaining expert tool.

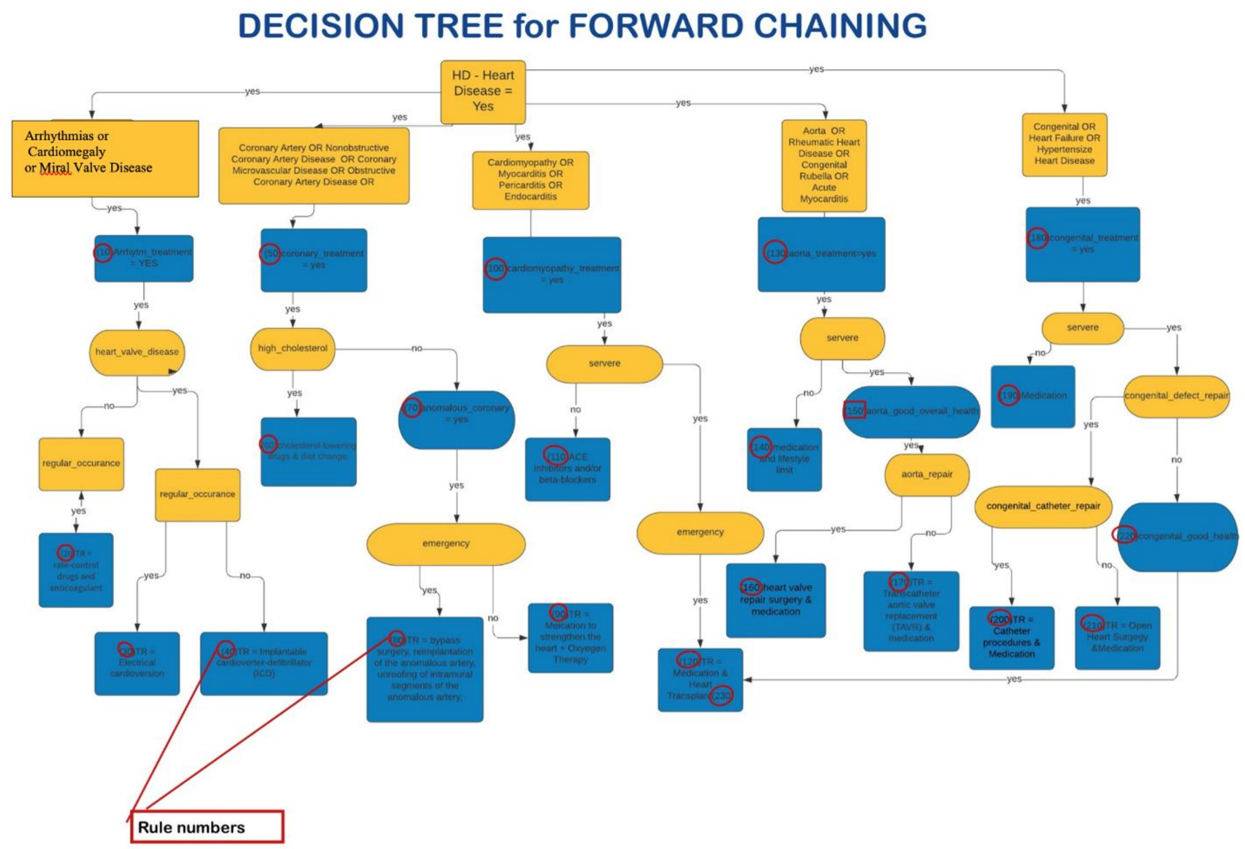
In this expert system I used the forward and backward chaining algorithms to properly assign the valid consequents to their corresponding rules based on a set of user queries output by the software. Predicated upon the user's response, the software is able to guide the user "backwards" through the knowledge base, to a generated diagnosis. This approach is considered backwards since it is goal driven. This fact can be more easily visualized by observing the backward chaining decision tree that I developed, which is documented further in this report. Next, I constructed the forward chaining decision tree which is derived from the diseases established in the backward chaining decision tree. This means that instead of predicating our decisions upon diagnostic conclusions, I started at the root of the tree (the diseases) and branched out based upon specific characteristics of the patient, who would ultimately be prescribed the proper treatment for their disease. Since it would allow me to have the components of the knowledge base (the diseases) required in order to properly construct the forward chaining tree, I developed the backward chaining decision tree before I developed the forward chaining decision tree. It is typical for the backward chaining algorithm to be implemented into a program before the forward chaining algorithm.

Decision Trees:

Backward Chaining:



Forward Chaining:


DECISION TREE for FORWARD CHAINING
HD - Heart Disease = Yes
yes
Arrhythmias or Cardiomegaly or Miral Valve Disease
Coronary Artery OR Nonobstructive Coronary Artery Disease OR Coronary Microvascular Disease OR Obstructive Coronary Artery Disease OR
Cardiomyopathy OR Myocarditis OR Pericarditis OR Endocarditis
Aorta OR Rheumatic Heart Disease OR Congenital Rubella OR Acute Myocarditis
Congenital OR Heart Failure OR Hypertensize Heart Disease
(10) Arrhytm_treatment = YES
(50) coronary_treatment = yes
(100) cardiomyopathy_treatment = yes
(130) aorta_treatment=yes
(180) congenital_treatment = yes
heart_valve_disease
high_cholesterol
servere
no
regular_occurance
(60) cholesterol-lowering drugs & diet change.
(70) anomalous_coronary = yes
(110) ACE inhibitors and/or beta-blockers
(140) medication and lifestyle limit
(150) aorta_good_overall_health
(190) Medication
congenital_defect_repair
aorta_repair
congenital_catheter_repair
(220) congenital_good_health
(20) TR = rate-control drugs and anticoagulant
emergency
(30) TR = Electrical cardioversion
(40) TR = Implantable cardioverter-defibrillator (ICD)
(80) TR = bypass surgery, reimplantation of the anomalous artery, unroofing of intramural segments of the anomalous artery.
(90) TR = Mecation to strengthen the heart + Oxygen Therapy
(120) TR = Medication & Heart Transplant (230)
(160) heart valve repair surgery & medication
(170) TR = Transcatheter aortic valve replacement (TAVR) & medication
(200) TR = Catheter procedures & Medication
(210) TR = Open Heart Surgegy &Medication
Rule numbers

By analyzing the above graphs, it is made simple to visualize and better understand the decision-making process that my program is processing. The graph comprised of yellow and blue ovals/rectangles is my forward chaining tree. The graph composed of white circles and blue rectangles is my backward chaining tree. These trees vary in size due to the project's requirement of at least 30 rules ("if-then" statements) to be included into the backward chaining algorithm. This requirement made the backward chaining decision tree significantly more detailed than the forward chaining tree.

Rules:

Backward Chaining Rules:

10 IF short_breath (SB) = NO THEN disease (DE) = NO
20 IF short_breath (SB) = YES and dizziness (DZ) = YES THEN balance_issue (BI) = YES
30 IF short_breath (SB) = YES and dizziness (DZ) = NO THEN stable (SA) = YES
40 IF balance_issue (BI) = YES and fatigue (FA) = YES and abnormal_heart_rhythm (AR) = YES THEN possible_cardiomegaly (PC) = YES
50 IF possible_cardiomegaly (PC) = YES and excess_iron_in_body (EI) = YES THEN disease (DE) = Cardiomegaly

60 IF possible_cardiomegaly (PC) = YES and excess_iron_in_body (EI) = NO THEN disease (DE) = Arrhythmias
70 IF balance_issue (BI) = YES and fatigue (FA) = YES and abnormal_heart_rhythm (AR) = NO THEN possible_heart_murmur (HM) = YES
80 IF possible_heart_murmur (HM) = YES and enlarged_neck_vien (NV) = YES THEN disease (DE) = Mitral Valve Disease
90 IF possible_heart_murmur (HM) = YES and enlarged_neck_vien (NV) = no and aching_joints_muscles (AM) = YES THEN possible_heart_muscle_disease (MD) = YES
100 IF possible_heart_muscle_disease (MD) = YES and fever (FE) = YES THEN disease (DE) = Myocarditis
110 IF possible_heart_muscle_disease (MD) = YES and fever (FE) = NO and pain_behind_brestbone (PB) = YES THEN disease (DE) = Pericarditits
120 IF possible_heart_muscle_disease (MD) = YES and fever (FE) = no and pain_behind_brestbone (PB) = NO THEN disease (DE) = Endocarditis
130 IF balance_issue (BI) = YES and fatigue (FA) = NO and nausea (NA) = YES THEN disease (DE) = Coronary Artery
140 IF balance_issue (BI) = YES and fatigue (FA) = NO and nausea (NA) = NO THEN swelling_test (ST) = YES
150 IF swelling_test (ST) = YES and swell_result (SR) = YES THEN disease (DE) = Cardiomyopathy
160 IF swelling_test (ST) = YES and swell_result (SR) = NO and sedentary_lifestyle (SL) =YES THEN possible_ischaemic_disease (ID) = YES
170 IF possible_ischaemic_disease (ID) = YES and high_cholesterol (HC) = YES THEN disease (DE) = Coronary Microvascular Disease
180 IF possible_ischaemic_disease (ID) = YES and high_cholesterol (HC) = NO THEN diagnostic_vascular_screening (VS) = YES
190 IF diagnostic_vascular_screening (VS) = YES and vascular_obstruction (VO) = YES THEN disease (DE) = Obstructive Coronary Artery Disease
200 IF diagnostic_vascular_screening (VS) = YES and vascular_obstruction (VO) = NO THEN disease (DE) = Non-obstructive Coronary Artery Disease
210 IF stable (SA) = YES and fainting (FN) = YES THEN disease (DE) = Aorta
220 IF stable (SA) = YES and fainting (FN) = NO THEN defect_issue (DI) = YES
230 IF defect_issue (DI) = YES and chest_pain (CP) = YES THEN disease (DE) = Congenital
240 IF defect_issue (DI) = YES and chest_pain (CP) = NO and high_blood_pressure (HP) = YES THEN possible_heart_failure (HF) = YES
250 IF possible_heart_failure (HF) = YES and thyroid_disease (TD) = YES THEN disease (DE) = Heart Failure
260 IF possible_heart_failure (HF) = YES and thyroid_disease (TD) = NO THEN disease (DE) = Hypertensive Heart Disease

270 IF defect_issue (DI) = YES and chest_pain (CP) = NO and high_blood_pressure (HP) = NO THEN viral_issue (VI) = YES
280 IF viral_issue (VI) = YES and scarlet_fever (SF) = YES THEN disease (DE) = Rheumatic Heart Disease
290 IF viral_issue (VI) = YES and scarlet_fever (SF) = NO and rubella_virus (RV) = YES THEN disease (DE) = Congenital Rubella
300 IF viral_issue (VI) = YES and scarlet_fever (SF) = NO and rubella_virus (RV) = NO and human_parvovirus_b19 (HB) = YES THEN disease (DE) = Acute Myocarditis

<u>Forward Chaining Rules:</u>
10 IF heart_diseases (HD) = YES and arrhythmias (AR) = YES THEN arrhytm_treatment (AT) = YES
20 IF arrhytm_treatment (AT) = YES and heart_valve_disease (HV) = NO and regular_occurance (RO) = YES THEN treatment (TR) = "rate-control drugs and anticoagulants"
30 IF arrhytm_treatment (AT) = YES and heart_valve_disease (HV) = YES and regular_occurance (RO) = YES THEN treatment (TR) = "electrical cardioversion"
40 IF arrhytm_treatment (AT) = YES and heart_valve_disease (HV) = YES and regular_occurance (RO) = NO THEN treatment (TR) = "Implantable cardioverter-defibrillator (ICD)"
50 IF heart_diseases (HD) = YES and coronary_artery (CA) = YES THEN coronary_treatment (CT) = YES
60 IF coronary_treatment (CT) = YES and high_cholesterol (HC) = YES THEN treatment (TR) = cholesterol-lowering drugs & diet change.
70 IF coronary_treatment (CT) = YES and high_cholesterol (HC) = NO THEN anomalous_coronary (AC) = YES
80 IF anomalous_coronary (AC) = YES and emergency (EM) = YES THEN treatment (TR) = bypass surgery, reimplantation of the anomalous artery, unroofing of intramural segments of the anomalous artery
90 IF anomalous_coronary (AC) = YES and emergency (EM) = NO THEN treatment (TR) = Medication to strengthen the heart + Oxygen Therapy
100 IF Heart diseases (HD) = YES and Cardiomyopathy (CY) = YES THEN cardiomyopathy_treatment (CT) = YES
110 IF cardiomyopathy_treatment (CT) = YES and severe (SE) = NO THEN treatment (TR) = "ACE inhibitors and/or beta-blockers"
120 IF cardiomyopathy_treatment (CT) = YES and severe (SE) = YES and emergency (EM) = YES THEN treatment (TR) = "Medication & Heart Transplant"
130 IF heart_diseases (HD) = YES and aorta (AO) = YES THEN aorta_treatment (OT) = YES
140 IF aorta_treatment (OT) = YES and severe (SE) = NO THEN treatment (TR) = "medication and lifestyle limit"

150 IF aorta_treatment (OT) = YES and severe (SE) = YES THEN aorta_good_overall_health (AG) = YES
160 IF aorta_good_overall_health (AG) = YES and aorta_repair (OR) = YES THEN treatment (TR) = "heart valve repair surgery & medication"
170 IF aorta_good_overall_health (AG) = YES and aorta_repair (OR) = NO THEN treatment (TR) = "Transcatheter aortic valve replacement (TAVR) & medication"
180 IF heart disease (HD) = YES and congenital (CN) = YES THEN congenital_treatment (GT) = YES
190 IF congenital_treatment (GT) = YES and severe (SE) = NO THEN treatment (TR) = "medication"
200 IF congenital_treatment (GT) = YES and severe (SE) = YES and congenital_defect_repair (GD) = YES and congenital_catheter_repair (GC) = YES THEN treatment (TR) = "Catheter procedures & Medication"
210 IF congenital_treatment (GT) = YES and severe (SE) = YES and congenital_defect_repair (GD) = YES and congenital_catheter_repair (GC) = NO THEN treatment (TR) = "Open Heart Surgery &Medication"
220 IF congenital_treatment (GT) = YES and severe (SE) = YES and congenital_defect_repair (GD) = NO THEN congenital_good_health (GG) = YES
230 IF congenital_good_health (GG) = YES THEN treatment (TR) = "Medication & Heart Transplant"

Program Implementation:
In order to efficiently store the knowledge base of my system, I implemented hash functions to map data of arbitrary size to a fixed value. That fixed value is then indexed inside my hash tables which serve as data structures that contain the hash elements. For this specific program the values that were stored are strings (i.e. antecedent variables, conclusions, ect.) instead of numbers. By using unordered maps and the string copy functions, I was able to successfully receive the valid string value and allocate it to a desired location in memory via backward and forward chaining structures. In order to completely implement these two algorithms, I used a combination of different data structures specific to each algorithm. To accurately implement the backward chaining algorithm into my program, I included the required data structures in the code such as a conclusion list, a conclusion stack, a variable list, and a clause variable list. The data that compromises these structures was stored using the aforementioned hash function technique. To properly implement the forward chaining algorithm into my program, I included the required data structures which are a clause variable list, a variable list, a conclusion variable queue, and a clause variable pointer. The data that compromises these structures was stored using the aforementioned hash function technique as well. Furthermore, I addressed the data accessing portion of the system by implementing the unordered map data structure into the program. An unordered map can be described as an associative container that contains

key-value pairs with unique keys in an unordered manner. We use these so-called keys to access their corresponding string variable stored in the hash table.

Analysis of the Program:

When analyzing the software, it is important to note that I used the sample code provided by my professor and debugged it to make it compile and run correctly. I essentially used this flawed program as a foundation for my own. While debugging this program I removed **many** errors. One of the most notable enhancements I made to the software was the removal of all of the "goto" statements in the code body. Though these statements were functional, the implementation of them in a program is generally frowned upon in the industry for various reasons, one of which being its inefficient time complexity. Once the code was debugged, I began dividing the jumble of human readable into different classes. I eventually decided that it was best to divide the code into two primary classes. One class would be comprised of all of the backward chaining related structures and elements. The second class would be comprised of all of the forward chaining related structures and elements. Once this was done I focused on enhancing the efficiency of the software by implementing the unorder map data structure and the accompanying hash elements to structure the storage and accessibility of the knowledge base. This data structure allowed the data of the knowledge base to be dynamically allocated into memory rather than statically. I then focused on further enhancing the efficiency and readability of the software by implementing certain aspects of traditional object-oriented principals. For instance, rather than including all of the code required to successfully compile and run the program in the main function, I divided it into separate and unique functions that were distinctly called either by one another or by the program's main function.

Analysis of the Results:

This software was inexecutable when I first received it. Simply by debugging the software I transformed useless code into a functioning expert system. Moreover, thanks to the knowledge base of this program being dynamically allocated, the knowledge base itself can be expanded without consequence. Without this newly added feature, complications that may lead the program to be too inefficient to function properly could occur. This enhancement legitimizes the efficacy of this expert system in the medical industry (or even for domestic use); although this software is relatively adolescent. The inclusion of object-oriented principals into the code body further developed the efficiency of the program and enhanced the readability of the human readable code. Implementing these principals also allow for unwitting software engineers, who aspire to develop an expert system, to more easily understand and contribute to this project without direct communication with its original developer. Generally, my modifications and newly added features to this software constructively enhanced its functionality, efficiency, and readability.

Conclusion:

What I concluded from this project is a greater comprehension of expert systems and the implementation process of the forward and backward chaining algorithms. In reflection, developing this system has provided me with a greater understanding and appreciation of the intricacies involved in programming an artificially intelligent system. I've digested the significant details about all of the primary components that make up my expert system. These components are the user query (the questions generated by our program), the user interface (the output of our executable file), the inference engine (in this case the backward and forward chaining algorithms), the knowledge base (all of the stored rules and their accompanying data), the response to the user (the disease diagnosis and treatment generated by our program), and all of the other, less pertinent, data structures encapsuled within my expert system that give it the ability to suitably function. Furthermore, I was required to rehash my comprehension of some C++ concepts and structures that I was not familiar with. Generally, this project has challenged me to become an improved critical thinker, software engineer, and artificial intelligent systems developer.

References:

The following hyperlinks can be used to view the sources that I used to develop the knowledge base for my expert system:

- https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118
- https://www.medicalnewstoday.com/articles/237191#types
- https://www.mayoclinic.org/diseases-conditions/heart-disease/diagnosis-treatment/drc-20353124