SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)

(Established under Section 3 of the UGC Act, 1956)

॥वसुधैव कुटुम्बकम्॥ Re-accredited by NAAC with 'A' grade (3.58/4) | Awarded Category – I by UGC

AI Evaluation – III

Medical Expert System- Covid-19

Submitted by –

Group 5

Tejas Kumar	**2203024055**
Nitish Jha	**22030242068**

Abstract

The provided prolog file is an Expert System for COVID-19 diagnosis. It allows the user to input symptoms, check symptoms, perform a temperature check, and perform a checkup to diagnose if the patient has COVID-19. The provided report highlights the main predicates of the program and highlights what is meant by each module.

Aim of the report is to provide the understanding of expert system and the predicates used in it. The report only has two sections, first section contains predicate and clauses with code snippets to provide the explanation, the second section showcases the exploration of the features provided by the ES.

The Expert System was derived from GitHub repository of Rojay White, and credits for the work goes to him. As per instructions mentioned in the Read.md of repository only Moh Project file was used.

Section 1: Predicates and Clauses Explanation

Post making the code executable for Prolog following 12 predicates were identified

dynamic(symptom/1).

In Prolog, dynamic/1 in Prolog provides a powerful mechanism for dynamically modifying the predicate database at runtime, which can be useful in a wide range of applications, including medical diagnosis systems, expert systems, and natural language processing systems. Its intended use in the program to interact with assert and retract systems. Which would have allowed it to

Symptom/1 predicate is the name of the predicate you want to make dynamic, and arity is the number of arguments the predicate takes.

For example, suppose you have a predicate called symptom/2 that represents a symptom and its severity. You can make this predicate dynamic using the dynamic/1 predicate like this:

dynamic (symptom/2).

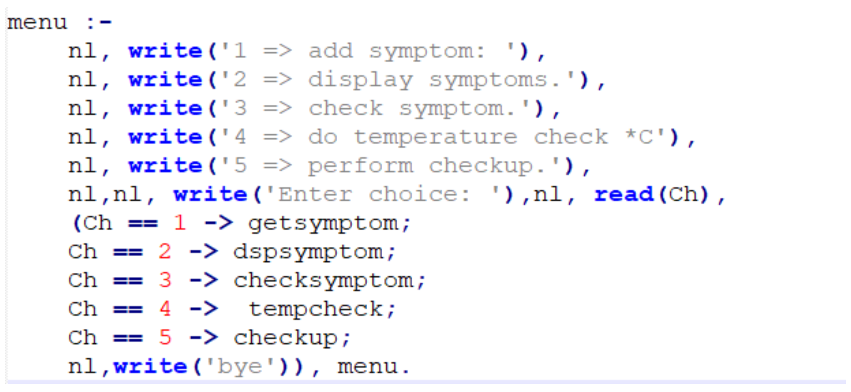
symptom(none).

It is not a predicate but a fact, which indicates that there are no recorded symptoms for the patient.

menu.

The menu clause in the program can be stated as a GUI component. In the given code, menu is a Prolog predicate that displays a list of options to the user and prompts for a choice. The user's choice is then used to execute a corresponding action by calling another predicate.

The menu predicate is defined using Prolog's built-in write and read predicates to display text and read input from the user. It uses a series of conditional statements (using ->) to check the user's choice and execute the appropriate action.

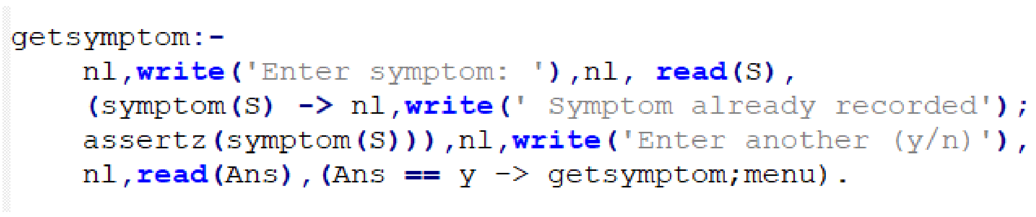


```
menu :-
    nl, write('1 => add symptom: '),
    nl, write('2 => display symptoms.'),
    nl, write('3 => check symptom.'),
    nl, write('4 => do temperature check *C'),
    nl, write('5 => perform checkup.'),
    nl,nl, write('Enter choice: '),nl, read(Ch),
    (Ch == 1 -> getsymptom;
    Ch == 2 -> dspsymptom;
    Ch == 3 -> checksymptom;
    Ch == 4 ->  tempcheck;
    Ch == 5 -> checkup;
    nl,write('bye')), menu.
```

The above snippet provides an overview of how the menu was defined in the code. We can see that it provides 5 features which will be explained in detail in the next section.

getsymptom

This procedure allows the user to add a new symptom to the symptom-checking system. Which as noticed previously is a feature in menu. It has been defined in program as follows:



```
getsymptom:-
    nl,write('Enter symptom: '),nl, read(S),
    (symptom(S) -> nl,write(' Symptom already recorded');
    assertz(symptom(S))),nl,write('Enter another (y/n)'),
    nl,read(Ans),(Ans == y -> getsymptom;menu).
```

The procedure first displays a message asking the user to enter a new symptom. The user enters the symptom by typing it in and pressing the Enter key. The procedure then checks if the symptom already exists in the system by calling the symptom/1 predicate. If the symptom is already present in the system, the procedure displays a message indicating that the symptom is already recorded.

If the symptom is not already present in the system, the procedure uses the assertz/1 predicate to add the symptom to the system. The assertz/1 predicate adds the new symptom to the end of the list of symptoms in the system.

After adding the symptom, the procedure displays a message asking the user if they want to enter another symptom. If the user enters 'y', the procedure calls itself recursively to allow the user to enter another symptom. If the user enters any input other than 'y', the procedure returns to the main menu by calling the menu/0 procedure.

assert_symptom(S)

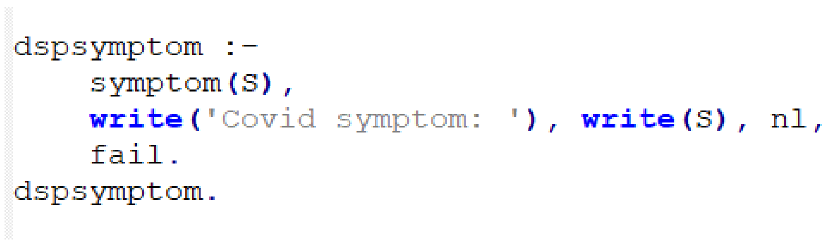
This predicate takes a symptom as its argument and uses the built-in assert/1 predicate to add the symptom to the symptom-checking system.

The assert/1 predicate adds a new fact to the database, which in this case is the symptom being asserted. The fact has the form symptom(S) where S is the symptom being asserted.

The assert_symptom/1 predicate can be called from other parts of the program to add a new symptom to the system. For example, if the user enters a new symptom using the getsymptom/0 procedure, the assert_symptom/1 predicate could be called to add the new symptom to the system.

dspsymptom

This procedure allows the user to display a list of all the symptoms currently in the symptom-checking system.The procedure first tries to match the symptom/1 predicate with a variable S. This effectively iterates over all the symptoms in the system, as each symptom is represented by a fact of the form symptom(S). It was written in following manner.



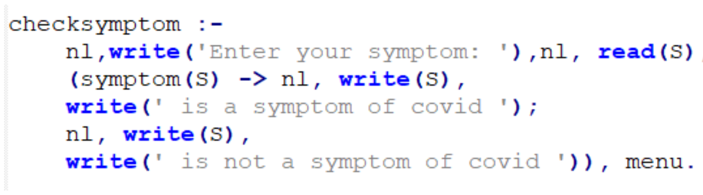
```
dspsymptom :-
    symptom(S),
    write('Covid symptom: '), write(S), nl,
    fail.
dspsymptom.
```

For each matching symptom/1 predicate, the procedure writes a message to the console indicating that the symptom is a Covid symptom, followed by the name of the symptom. The nl predicate is used to print a newline character after each symptom.

The fail/0 predicate is then called to fail and backtrack to the previous symptom/1 predicate, effectively continuing the iteration over all the symptoms. This ensures that all symptoms are printed to the console.

Finally, when there are no more matching symptom/1 predicates, the procedure backtracks to the dspsymptom/0 predicate definition and succeeds, completing the procedure.

checksymptom



```
checksymptom :-
    nl,write('Enter your symptom: '),nl, read(S),
    (symptom(S) -> nl, write(S),
    write(' is a symptom of covid ');
    nl, write(S),
    write(' is not a symptom of covid ')), menu.
```

This procedure allows the user to check if a particular symptom is present in the symptom-checking system and if it is a symptom of Covid.

The procedure first displays a message asking the user to enter a symptom. The user enters the symptom by typing it in and pressing the Enter key. The symptom is stored in the variable S.

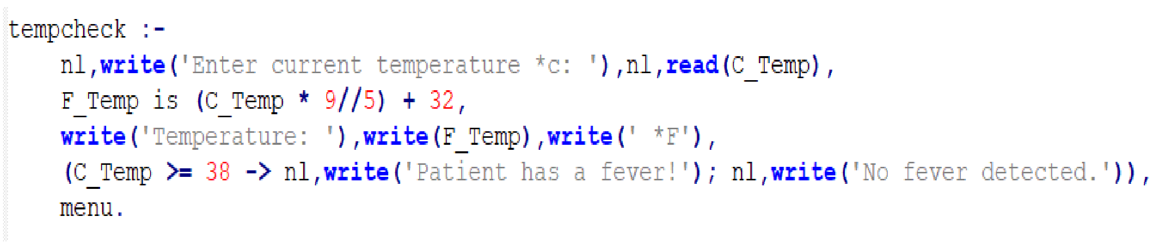
The procedure then checks if the symptom exists in the system by calling the symptom/1 predicate. If the symptom is present in the system, the procedure displays a message indicating that the symptom is a symptom of Covid. If the symptom is not present in the system, the procedure displays a message indicating that the symptom is not a symptom of Covid.

Finally, the procedure returns to the main menu by calling the menu/0 procedure.

tempcheck

This code snippet is written in Prolog and defines a procedure called tempcheck. This procedure allows the user to enter their current body temperature in degrees Celsius and converts it to degrees Fahrenheit. It then checks if the temperature is above 38 degrees Celsius, which is considered a fever.

The procedure first displays a message asking the user to enter their current body temperature in degrees Celsius. The user enters the temperature by typing it in and pressing the Enter key. The temperature is stored in the variable C_Temp.



```
tempcheck :-
    nl,write('Enter current temperature *c: '),nl,read(C_Temp),
    F_Temp is (C_Temp * 9//5) + 32,
    write('Temperature: '),write(F_Temp),write(' *F'),
    (C_Temp >= 38 -> nl,write('Patient has a fever!'); nl,write('No fever detected.')),
    menu.
```

The procedure then converts the temperature from degrees Celsius to degrees Fahrenheit using the formula F_Temp = (C_Temp * 9//5) + 32, where // is the integer division operator. The result is stored in the variable F_Temp.

The procedure then displays a message indicating the converted temperature in degrees Fahrenheit.

The procedure then checks if the temperature is above 38 degrees Celsius by comparing it to the value 38. If the temperature is above 38 degrees Celsius, the procedure displays a message indicating that the patient has a fever. If the temperature is not above 38 degrees Celsius, the procedure displays a message indicating that no fever is detected.

Finally, the procedure returns to the main menu by calling the menu/0 procedure.

checkup

This procedure allows the user to perform a symptom-checking system to determine if they have any diseases.

The procedure first calls the checkFor/1 predicate to determine if the user has any diseases based on their symptoms. The checkFor/1 predicate is not shown in this code snippet, but it is assumed to be defined elsewhere in the program. Please refer following for syntax.

```
checkup :-
    checkFor(Disease),
    nl,write('Processing....'),nl,
    write('System check......patient has: '),
    write(Disease),nl,
    undo,
    nl,write('**********************System Close**********************').
```

After determining the user's disease, the procedure displays a message indicating that the system is processing the results. The procedure then displays a message indicating that the system has checked the patient and indicates the disease that the patient has.

The procedure then calls the undo/0 predicate to remove any previously asserted symptoms from the symptom-checking system. This is done so that the symptom-checking system can be reset for the next patient.

Finally, the procedure displays a message indicating that the symptom-checking system is closed and returns to the main menu.

covid

The covid predicate checks the symptoms associated with Covid and gives advice to the user based on their symptoms. It was defined as follows

```
covid :-
    checkSymptom(headache),
    checkSymptom(fever),
    checkSymptom(dry_cough),
    checkSymptom(body_ache),
    checkSymptom(chest_pain),
    checkSymptom(shortnes_of_breath),
    checkSymptom(loss_of_speach_or_movement),
    write('Based on the symptoms, the patient may have COVID-19.'), nl,
    write('Advice: '),nl,
    write('_____Short Term:______ .'), nl,
    nl,write('- wear a mask.'),
    nl,write('- drink plenty of fluids.'),
    nl,write('- use hand sanitizer.'),nl,
    nl,write('______Long Term:______ '),
    nl,write('- vaccinate.'),
    nl,write('- self isolate.'),
    retractall(yes(_)),
    retractall(no(_)).
```

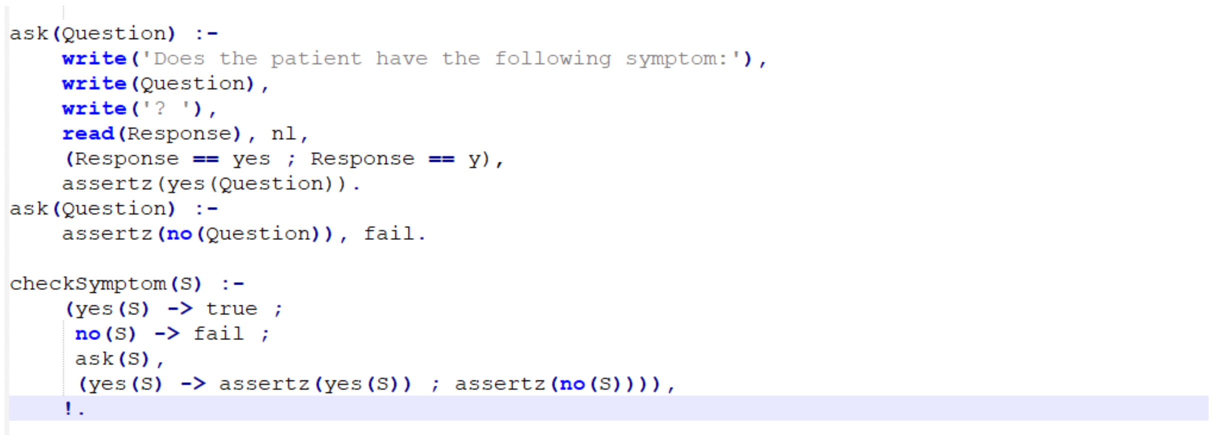
The covid predicate first checks for a list of symptoms associated with Covid by calling the checkSymptom/1 predicate for each symptom. The checkSymptom/1 predicate is assumed to be defined elsewhere in the program.

After checking the symptoms, the predicate prompts the user to enter any additional symptoms they may have and asserts them using the yes/1 predicate. This allows the symptom-checking system to be updated with new symptoms.

The predicate then gives advice to the user based on their symptoms. The advice is divided into short-term and long-term sections, with recommendations for things like wearing a mask, drinking fluids, and self-isolating in the short term, and getting vaccinated and seeking medical attention in the long term.

It is worth noting that the advice given by this predicate is very general and should not be taken as medical advice. Anyone experiencing symptoms of Covid, or any other illness should seek advice from a qualified medical professional.

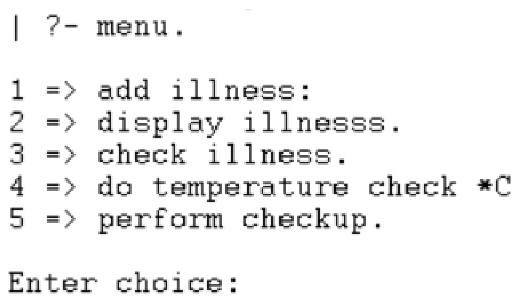
checksymptom



```
ask(Question) :-
    write('Does the patient have the following symptom:'),
    write(Question),
    write('? '),
    read(Response), nl,
    (Response == yes ; Response == y),
    assertz(yes(Question)).
ask(Question) :-
    assertz(no(Question)), fail.

checkSymptom(S) :-
    (yes(S) -> true ;
     no(S) -> fail ;
     ask(S),
     (yes(S) -> assertz(yes(S)) ; assertz(no(S)))),
    !.
```

As the above code indicates it uses the symptoms highlighted in covid predicate, if user enters y then it is recorded as true else it fails. This helps the expert system to determine whether patient has Covid if all the symptoms match it provides advise that is defined in covid clause. The connection is made using assert.

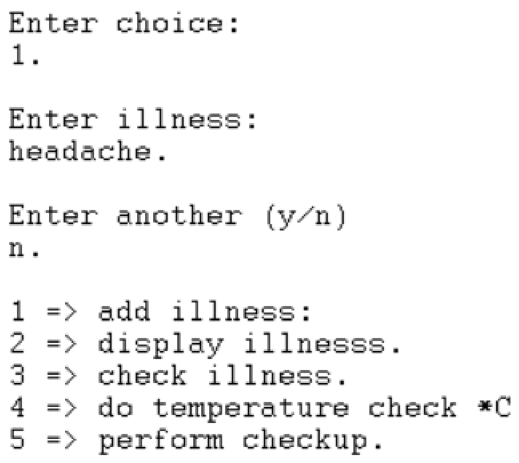
Section 2: Feature Demonstration



```
| ?- menu.

1 => add illness:
2 => display illnesss.
3 => check illness.
4 => do temperature check *C
5 => perform checkup.

Enter choice:
```

Menu as defined earlier is a UI feature which provides user with 5 options from ES which are mentioned in the above image. Each of the option will now be traversed and explained in following part of report.

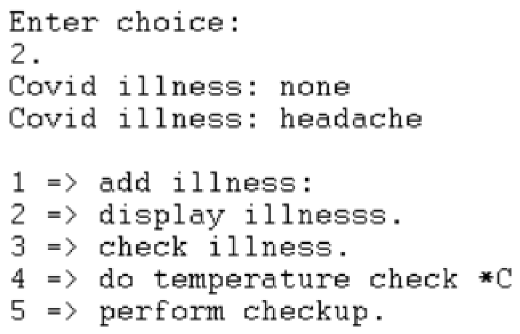
1st Feature



```
Enter choice:
1.

Enter illness:
headache.

Enter another (y/n)
n.

1 => add illness:
2 => display illnesss.
3 => check illness.
4 => do temperature check *C
5 => perform checkup.
```

This feature prompts user for the illness which they may be having and it keeps asking until they type n. The user can add any illness and it will be recorded by the system however the n, which user provided is recorded as none. As we have added fact Symptom(none).

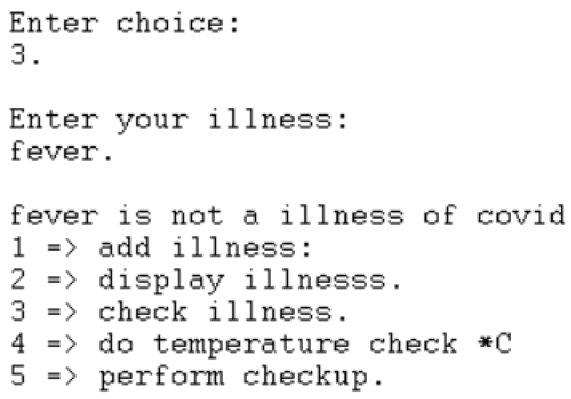
2nd Feature



```
Enter choice:
2.
Covid illness: none
Covid illness: headache

1 => add illness:
2 => display illnesss.
3 => check illness.
4 => do temperature check *C
5 => perform checkup.
```

This feature displays the illness which were provided by the user. As explained earlier it would not hide the input ‘n’ but will take it as a new input symptom (None). That is why wew can notice it gives output as Covid illness: none.

3rd Feature



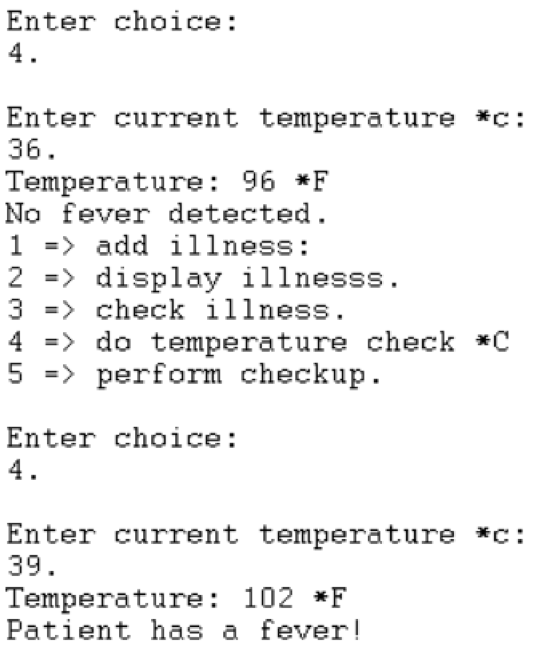
```
Enter choice:
3.

Enter your illness:
fever.

fever is not a illness of covid
1 => add illness:
2 => display illnesss.
3 => check illness.
4 => do temperature check *C
5 => perform checkup.
```

This feature again prompts user for illness input, and it checks it directory and prompts whether the illness of covid.

4th Feature

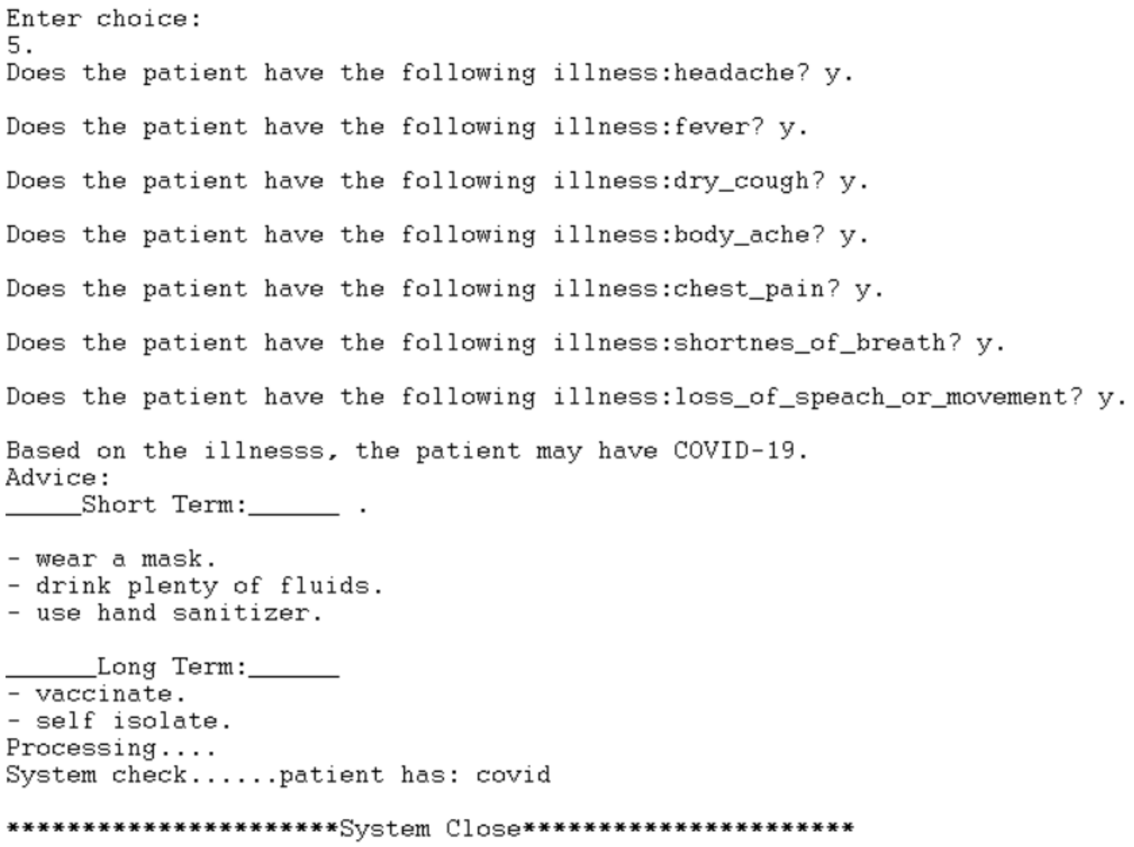
Image provided in next page*

This feature prompts user for temperature. Note the temperature is recorded in degrees Celsius, it then converts it to degree Fahrenheit, and also if it is greater than 38 degrees it highlights that user has Fever.



```
Enter choice:
4.

Enter current temperature *c:
36.
Temperature: 96 *F
No fever detected.
1 => add illness:
2 => display illnesss.
3 => check illness.
4 => do temperature check *C
5 => perform checkup.

Enter choice:
4.

Enter current temperature *c:
39.
Temperature: 102 *F
Patient has a fever!
```

5th Feature



```
Enter choice:
5.
Does the patient have the following illness:headache? y.

Does the patient have the following illness:fever? y.

Does the patient have the following illness:dry_cough? y.

Does the patient have the following illness:body_ache? y.

Does the patient have the following illness:chest_pain? y.

Does the patient have the following illness:shortnes_of_breath? y.

Does the patient have the following illness:loss_of_speach_or_movement? y.

Based on the illnesss, the patient may have COVID-19.
Advice:
_____Short Term:______ .

- wear a mask.
- drink plenty of fluids.
- use hand sanitizer.

______Long Term:______
- vaccinate.
- self isolate.
Processing....
System check......patient has: covid

**********************System Close**********************
```

The fifth feature prompts user for multiple questions and if all questios are answered y than it classifies condition as covid and gives advise in manner highlighted above.