NAME - SHIVAM.

CLASS - B.TECH (CS) 3rd year

ROLL NO - 1913103

SUBJECT - Artificial Intelligence And Machine learning.

191303
Bhavam

Assignment II

Ques 1. Write down logical representations for following :-

- Horses, cows and pigs are mammals
  Horse (x) ⇒ Mammal (x)
  Cow (x) ⇒ Mammal (x)
  Pig (x) ⇒ Mammal (x).

- An offspring of a horse is a horse
  Offspring (x, y) ∧ Horse (y) ⇒ Horse (x)

- Bluebeard is a horse
  Horse (Bluebeard)

- Bluebeared is Charlie's parent
  Parent (Bluebeard, Charlie).

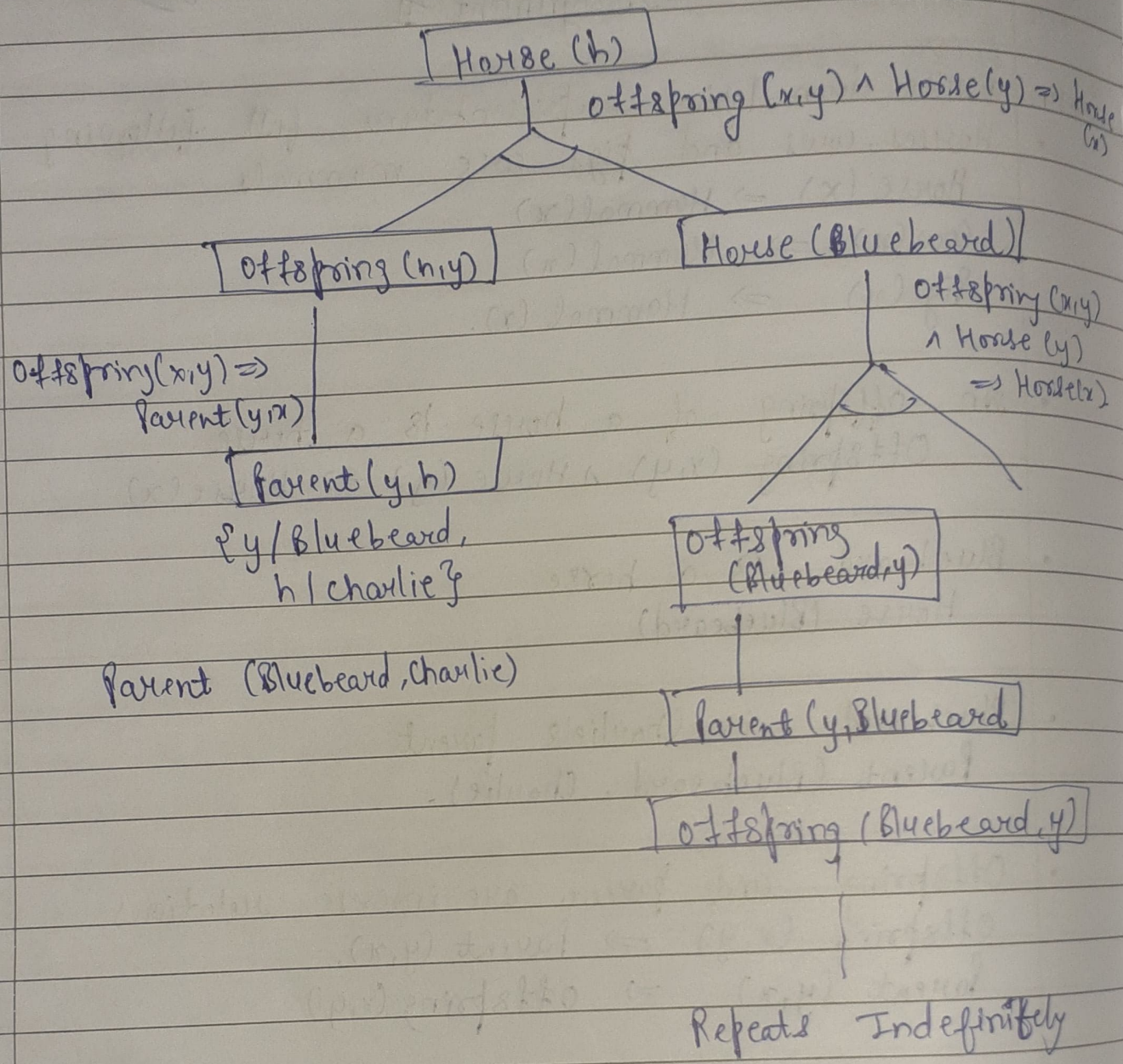
- Offspring and parent are inverse relations
  Offspring (x, y) ⇒ Parent (y, x).
  Parent (y, x) ⇒ offspring (x, y)

- Every Mammal has a parent
  Mammal (x) ⇒ HasParent (x)

Draw the proof tree generated by exhaustive backward chaining algorithm for query ∃h Horse(h), where clauses are matched in order given

191305
Bhavam

1913107
Bhuvan



1913107
Bhuvan

1918103 Shivam

Ques-2

(a) Translate these sentences into formulas in Predicate logic.

- John likes all kinds of food.
  $\forall x$ food(x) $\Rightarrow$ Like (John, x).

- Apples are food
  food (apples)

- Chicken is food
  food (chicken)

- Anything anyone eats and isn't killed by is food
  $\forall x \forall y$ eats (y, x) $\wedge$ alive (y) $\Rightarrow$ food(x)

- Bill eats peanuts and is still alive
  ~~$\forall x$, eats (Bill, x) $\Rightarrow$ eats~~
  eats (Bill, Peanuts) $\wedge$ alive (Bill).

- Sue eats everything Bill eats
  $\forall x$ eats (Bill, x) $\Rightarrow$ eats (Sue, x).

(b) Prove that John likes peanuts using backward chaining

$\forall x$ food (x) $\Rightarrow$ like (John, x) — (1)
food (apples) — (2)
food (chicken) — (3)
$\forall x \forall y$ eats (y, x) $\wedge$ alive (y) $\Rightarrow$ food(x) — (4)
eats (Bill, Peanuts) $\wedge$ alive (Bill) — (5)
$\forall x$ eats (Bill, x) $\Rightarrow$ eats (Sue, x) — (6)

1918103 Shivam

19/3/07 Shivam

Page No.: 4
Date:

In Backward chaining, we will start with our goal predicate which is Likes (John, Peanuts)

St 1 Likes (John, Peanuts)

Using eq ① ,

Likes (John, Peanuts)
|
food (Peanuts) {Peanuts/x}.

Now, using eq ②.

eats (y, Peanuts) ∧ alive (y)

Likes (John, Peanuts)
↑
food (Peanuts) {Peanuts/x}

eats (y, Peanuts) alive (y) {Bill/y}.
| ↑
eats (Bill, Peanuts) alive (Bill)

So, Proved ⇒ John likes Peanuts.

(c) Convert formulars into clause form

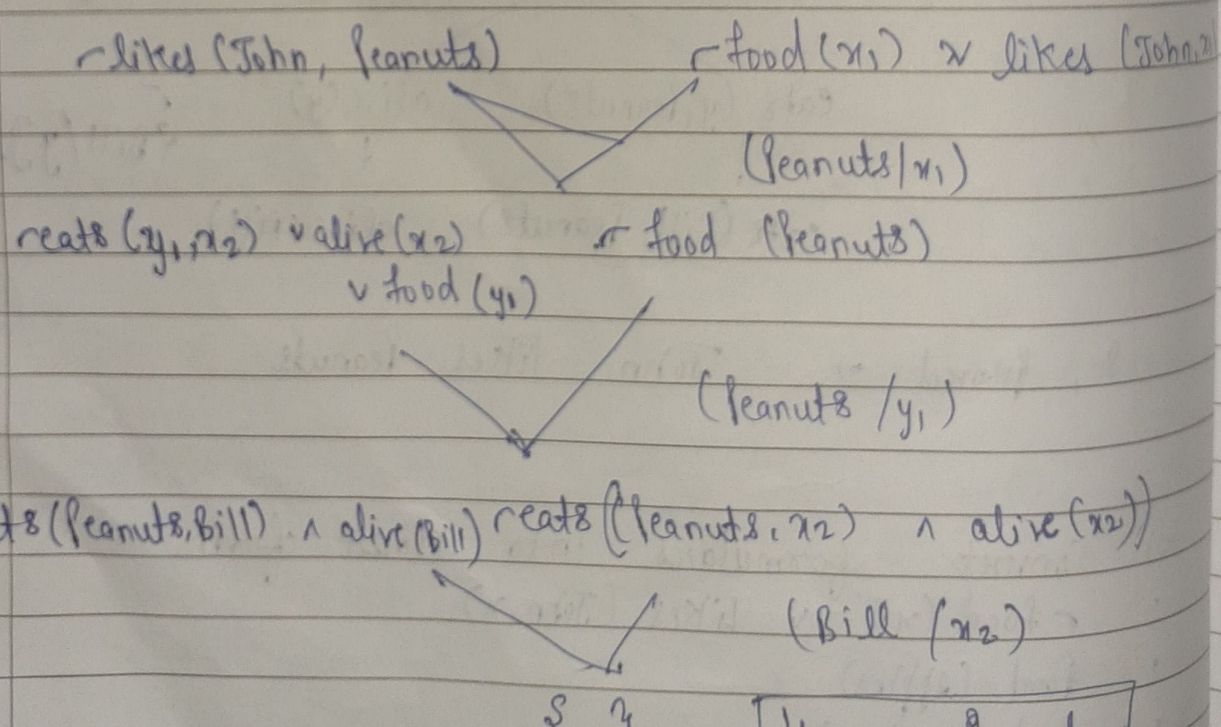
- ¬food (x) ∨ likes (John, x)
- food (Apples)
- food (chicken)

19/3/07 Shivam

- ¬eats (y, x) ∨ alive (y) ∨ food (x)
- eats (Bill, Peanuts)
- alive (Bill)
- ¬eats (Bill, x) ∨ eats (Sue, x).

(d) Prove that John likes peanuts using resolution
Rename variables:
¬food(x1) ∨ likes (John, x1)
food (apples)
food (chicken)
¬eats (y1, x2) ∨ alive (x2) ∨ food(y1)
eats (Bill, Peanuts) ∧ alive (Bill)
¬eats (Bill, x3) ∨ eats (Sue, x3).

Prove: "John likes Peanuts".
¬likes (John, Peanuts)



(e) Use resolution to answer: "What food does Sue eat"?

Sol To prove: eats (Sue, x)

¬eats (Sue, x) ¬eats (Bill, x) ∨ eats (Bill, x) [Sue written above Bill]

¬eats (Bill, x) eats (Bill, Peanuts)

(Peanuts/x)

{ }

So, Sue eats Peanuts.
eats (Sue, Peanuts)

Ques-3

(a) Represent facts in predicate logic.

- ~~Member (Elm St. Bridge Club, Joe)~~
- member (Joe, Club)
- member (Sally, Club)
- member (Bill, Club)
- member (Ellen, Club)
- marriedTo (Joe, Sally)
- Brother (Ellen, Bill)
- marriedTo (x, y) ∧ memberOf (x, Club) → memberOf (y, Club)
- lastMeeting (Club, houseOf(Joe))

Page No.: 7
Date: YOUVA

⇒ We can't give resolution proof for 'the last meeting of club was at Sally's house' because there is no rule which indicates that married couple live in same house or indicates that any event occuring in house of a married person also occeurs in his/her spouse.

Rule: MarriedTo (x,y) ∧ lastMeetingAt (Club, houseof(x)) → lastMeetingAt (club, Houseof(y)) — ⑦

⇒ We can't establish proof for "Ellen is not Married" because there is no rule indicating that relation and members in club and being unmarried

Rule:- member (x), member (y), ¬married (x,y) → ¬MarriedPerson (y) — ⑩

To Prove:- last meeting of Club was at Sally's house
Goal:- lastMeetingAt (Club, Houseof (Sally))

Equations:-
member (Club, Joe) — ①
member (Club, Sally) — ②
member (club, Bill) — ③
member (Club, Ellen) — ④
marriedTo (Joe, Sally) — ⑤
Brother (Ellen, Bill) — ⑥

$\text{marriedTo}(x,y) \wedge \text{memberOf}(x,\text{club}) \rightarrow \text{memberOf}(y,\text{club})$ — (7)

$\text{lastMeetingAt}(\text{club}, \text{HouseOf}(\text{Joe}))$ — (8)

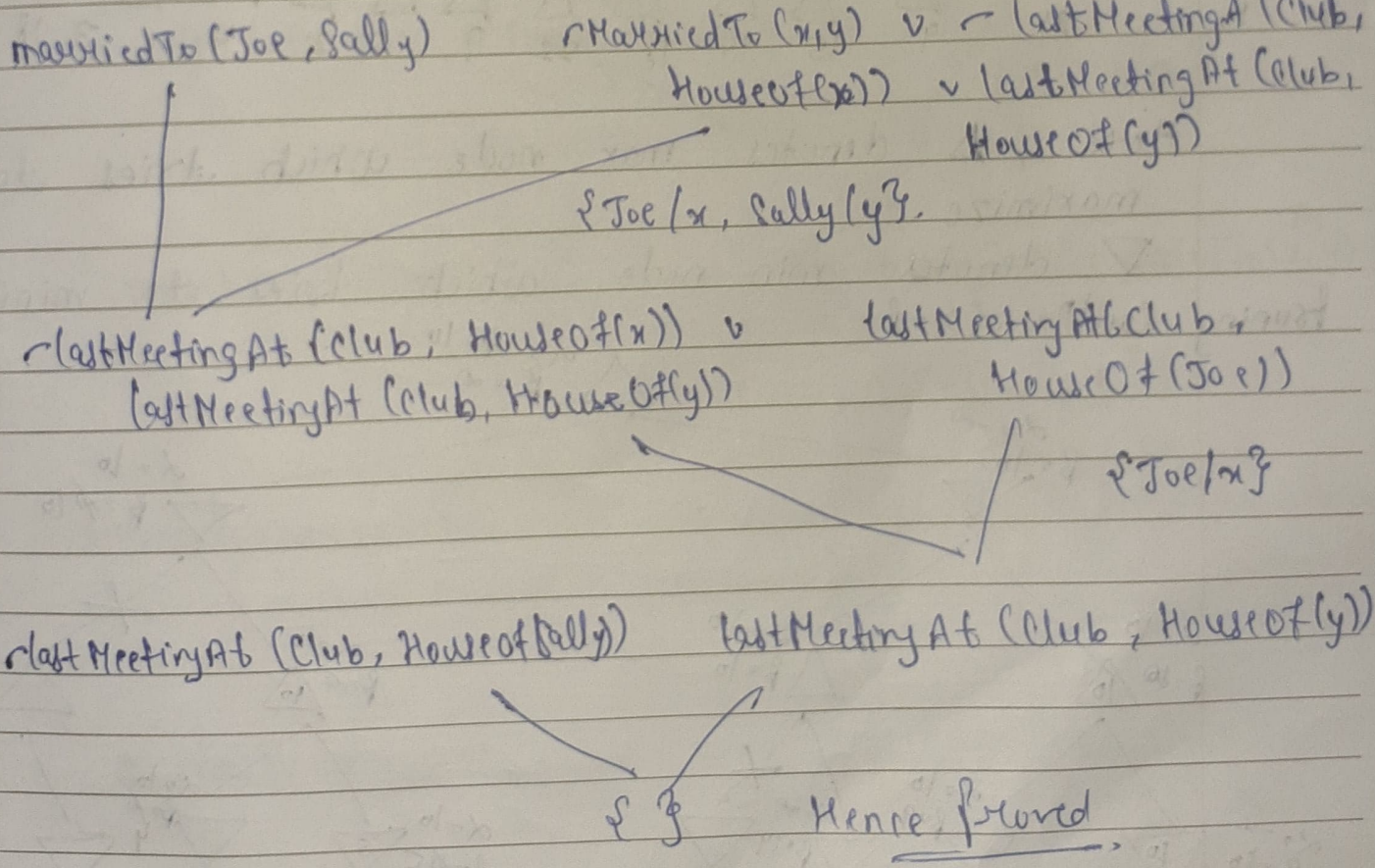
Convert eq (7) into clause form

$\neg \text{MarriedTo}(x,y) \vee \neg \text{lastMeetingAt}(\text{Club}, \text{HouseOf}(x)) \vee \text{lastMeetingAt}(\text{Club}, \text{HouseOf}(y))$ — (9)

Negate the goal

$\neg \text{lastMeetingAt}(\text{Club}, \text{HouseOf}(\text{Sally}))$ — (10)

from (5) and (9),



Hence proved.

19/3/03 Shivam

Page No.: 9
Date:

Prove that Ellen is not Married.

Goal: ¬MarriedPerson(Ellen)

Convert eq ⑥ into clause form.

¬Member(x) ∨ ¬Member(y) ∨ married(x,y) ∨ ¬MarriedPerson(y) — (10')

Negate the goal

MarriedPerson(Ellen) — (14)

From eq ① & (10'),

¬Member(x) ∨ ¬Member(y) ∨ marriedTo(x,y) ∨ ¬MarriedPerson(y)

member(Joe) {Joe/x}

member(Ellen) {Ellen/y}

¬member(y) ∨ marriedTo(Joe, y) ∨ ¬MarriedPerson(y)

MarriedPerson(Ellen)

marriedTo(Joe, Ellen) ∨ ¬MarriedPerson(Ellen)

MarriedTo(Joe, Sally)

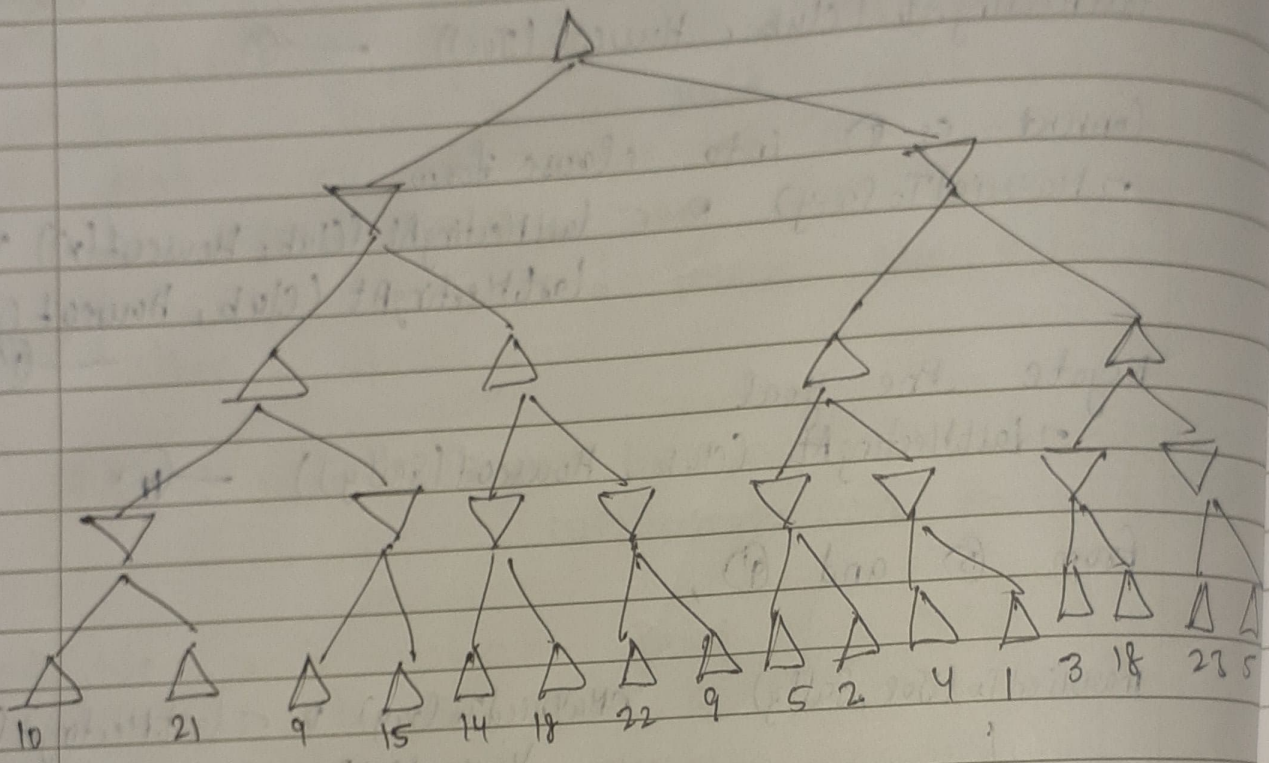
marriedTo(Joe, Ellen) { }

Hence proved

⇒ Ellen is not Married.

19/3/03 Shivam

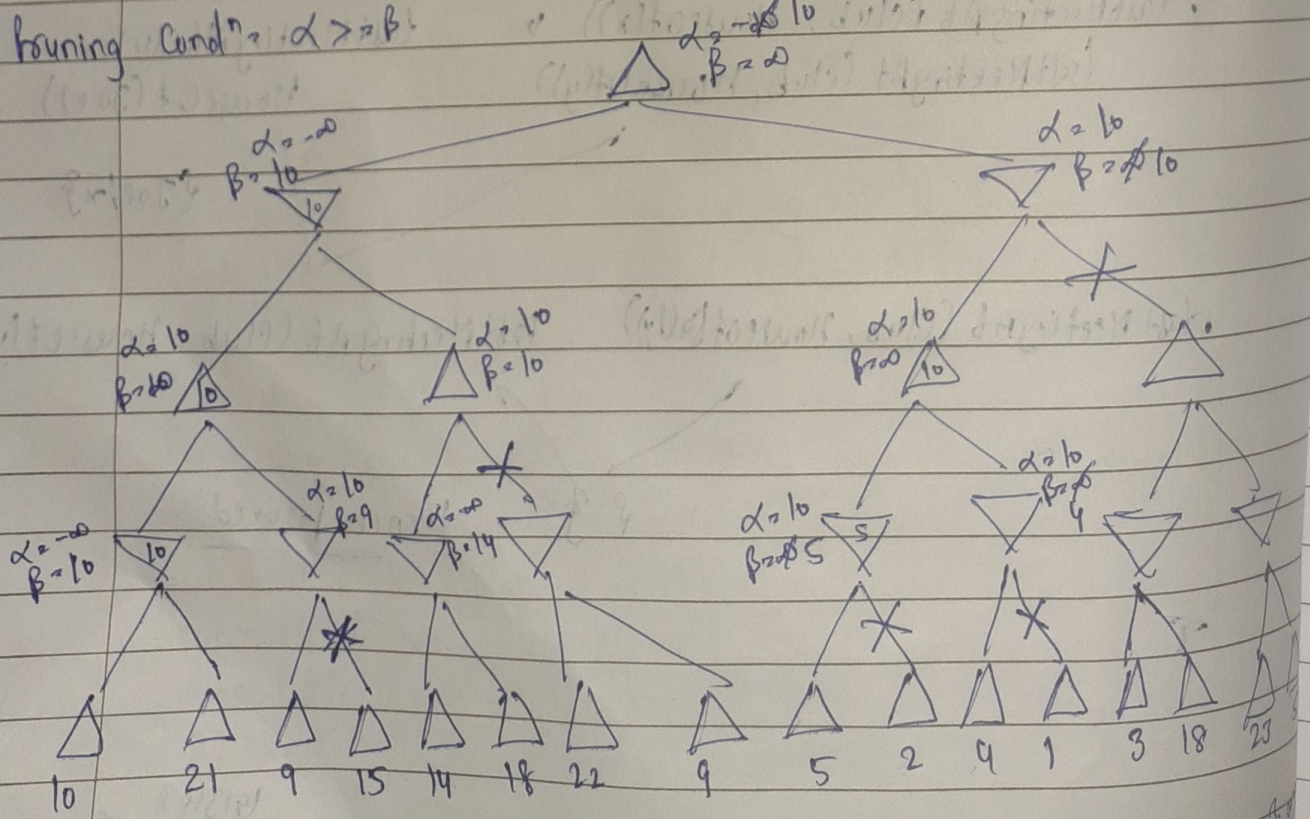
Ques-4 Apply Alpha-Beta pruning on following example:-



Here, △ denotes max node which tries to maximize

▽ denotes min node which tries to minimize

Pruning Cond^n :- $\alpha >= \beta$.



1918103. Bhuvan

Ques

Ans A* algorithm is a searching algorithm that searches for a shortest path between initial and final state.

function of A* algorithm

$f(n) = g(n) + h(n)$ where

$g(n)$ = cost to reach node n from start state.

$h(n)$ = cost from node n to goal node

$f(n)$ = total cost of cheapest solution.

As the three conditions given in question I disagree with the statement that A* algorithm may not terminate with shortest path.

A* is complete if:

=> Branching factor is finite.

=> costs of each edge is greater than zero

=> Heuristic function underestimates the distance to the goal.

This property is known as the admissibility of A*.

An admissible heuristic is optimistic in nature.

If heuristic function is admissible, then A* tree search will find least cost path.

1918103. Bhuvan