AI Homework 2

Ahmed Burak Gulhan
150160903

Problem 1)

1)All cats are animal.

$\forall x \, (cat(x) \Rightarrow animal(x))$

2)Everyone who owns a car also has a bicycle.

$\forall x \, (own\_car(x) \Rightarrow has\_bicycle(x))$

3)There is a student registered to AI class who talks to other students that are registered to AI class.

$\exists x \, \exists y \, (Registered(x, AI) \wedge Talks(x,y) \wedge Registered(y, AI))$

Problem 2)

- English, French, Russian and Turkish belong to language category (Same as ”English, French, Russian and Turkish are languages”)

  ∀x English(x) ⇒ Language(x)
  ∀x French(x) ⇒ Language(x)
  ∀x Russian(x) ⇒ Language(x)
  ∀x Turkish(x) ⇒ Language(x)

- Arda, Cihan and Gamze all are students in the same university.

  Student(Arda)
  Student(Cihan)
  Student(Gamze)
  ∃x (University(x) ∧ Enrolled(Arda,x) ∧ Enrolled(Cihan,x) ∧ Enrolled(Gamze,x))

- Each student in the university speaks at least two languages.

  ∀x ∃y ∃z (Student(x) ∧ University(x) ⇒ Speak(x, Language(y)) ∧ Speak(x, Language(z)) ∧ (y≠z)

- Fish and hamburger belong to food category.

  ∀x Fish(x) ⇒ Food(x)
  ∀x Hamburger(x) ⇒ Food(x)

- Classic, jazz and rock belong to music category.

  ∀x Classic(x) ⇒ Music(x)
  ∀x Jazz(x) ⇒ Music(x)
  ∀x Rock(x) ⇒ Music(x)

- Students who speak French like jazz music and dislike rock music.

  ∀x (Student(x) ∧ Speaks(x, French) ⇒ Like(x, Jazz) ∧ ¬Like(x, Rock))

- All students who speak Russian like rock music.

  ∀x (Student(x) ∧ Speaks(x, Russian) ⇒ Like(x, Rock))

- All students who like hamburger speak English but do not speak Turkish.

  ∀x (Student(x) ∧ Like(x, Hamburger) ⇒ Speak(x, English) ∧ ¬Speak(x, Turkish))

- All students who like fish and classic music speak Turkish.

∀x (Student(x) ∧ Like(x, Fish) ∧ Like(x, Classic) ⇒ Speak(x, Turkish))

- Arda likes jazz music, hamburger, fish but dislikes classic music, rock music.

  Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda, Classic) ∧ ¬Like(Arda, Rock)

- Cihan dislikes whatever music Arda likes, and he likes whatever music Arda dislikes. He likes hamburger and dislikes fish.

  ∀x (Music(x) ∧ Like(Arda,x) ⇒ ¬Like(Cihan, x))
  ∀x (Music(x) ∧ ¬Like(Arda,x) ⇒ Like(Cihan, x))
  Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)

- Gamze likes fish, classic music but dislikes hamburger, jazz music and rock music.

  Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)

a) Knowledge Base

Non CNF knowledge base:

∀x English(x) ⇒ Language(x)
∀x French(x) ⇒ Language(x)
∀x Russian(x) ⇒ Language(x)
∀x Turkish(x) ⇒ Language(x)
Student(Arda)
Student(Cihan)
Student(Gamze)
∃x (University(x) ∧ Enrolled(Arda,x) ∧ Enrolled(Cihan,x) ∧ Enrolled(Gamze,x))
∀x ∃y ∃z (Student(x) ∧ University(x) ⇒ Speak(x, Language(y)) ∧ Speak(x, Language(z)) ∧ (y≠z)
∀x Fish(x) ⇒ Food(x)
∀x Hamburger(x) ⇒ Food(x)
∀x Classic(x) ⇒ Music(x)
∀x Jazz(x) ⇒ Music(x)
∀x Rock(x) ⇒ Music(x)
∀x (Student(x) ∧ Speaks(x, French) ⇒ Like(x, Jazz) ∧ ¬Like(x, Rock))
∀x (Student(x) ∧ Speaks(x, Russian) ⇒ Like(x, Rock))
∀x (Student(x) ∧ Like(x, Hamburger) ⇒ Speak(x, English) ∧ ¬Speak(x, Turkish))
∀x (Student(x) ∧ Like(x, Fish) ∧ Like(x, Classic) ⇒ Speak(x, Turkish))
Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda, Classic) ∧ ¬Like(Arda, Rock)
∀x (Music(x) ∧ Like(Arda,x) ⇒ ¬Like(Cihan, x))
∀x (Music(x) ∧ ¬Like(Arda,x) ⇒ Like(Cihan, x))
Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)
Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)

Convert knowledge base to CNF:

1) ¬English(x) ∨ Language(x)
2) ¬French(x) ∨ Language(x)
3) ¬Russian(x) ∨ Language(x)
4) ¬Turkish(x) ∨ Language(x)
5) Student(Arda)
6) Student(Cihan)
7) Student(Gamze)
8) University(G(x)) ∧ Enrolled(Arda,G(x)) ∧ Enrolled(Cihan,G(x)) ∧ Enrolled(Gamze,G(x))
9) ¬Student(x) ∨ ¬University(x) ∨ (Speak(x, Language(G_1(x))) ∧ Speak(x, Language(G_2(x))) ∧ (y≠z))
10) ¬Fish(x) ∨ Food(x)
11) ¬Hamburger(x) ∨ Food(x)
12) ¬Classic(x) ∨ Music(x)
13) ¬Jazz(x) ∨ Music(x)
14) ¬Rock(x) ∨ Music(x)
15) [¬Student(x) ∨ ¬Speaks(x, French) ∨ (Like(x, Jazz)] ∧ [¬Student(x) ∨ ¬Speaks(x, French) ∨ ¬Like(x, Rock)]
16) ¬Student(x) ∨ ¬Speaks(x, Russian) ∨ (Like(x, Rock))
17) [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ Speak(x,English)] ∧ [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ ¬Speak(x,Turkish)]
18) ¬Student(x) ∨ ¬Like(x, Fish) ∨ ¬Like(x, Classic) ∨ (Speak(x, Turkish))
19) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock)
20) ¬Music(x) ∨¬ Like(Arda,x) ∨ (¬Like(Cihan, x))
21) ¬Music(x) ∨ Like(Arda,x) ∨ (Like(Cihan, x))
22) Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)
23) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)

b) Use resolution to answer the following queries:

1) Which student(s) speak(s) French

With the resolution algorithm we can only determine whether a query is True or False. Therefore for this query, (which expects a group of students as an answer) we have to determine whether each student can speak French seperately.

Students = {Arda, Cihan, Gamze}

∀x (Student(x) ∧ Speak(x, French)) Which contains the queries:

Speak(Arda, French), Speak(Cihan, French), Speak(Gamze, French)

Using Proof by Contradiction for query Speak(Arda, French)

1) ¬English(x) ∨ Language(x)
2) ¬French(x) ∨ Language(x)
3) ¬Russian(x) ∨ Language(x)
4) ¬Turkish(x) ∨ Language(x)
5) Student(Arda)
6) Student(Cihan)
7) Student(Gamze)
8) University(G(x)) ∧ Enrolled(Arda,G(x)) ∧ Enrolled(Cihan,G(x)) ∧ Enrolled(Gamze,G(x))
9) ¬Student(x) ∨ ¬University(x) ∨ (Speak(x, Language(G_1(x))) ∧ Speak(x, Language(G_2(x))) ∧ (G_1(x)≠G_2(x)))
10) ¬Fish(x) ∨ Food(x)
11) ¬Hamburger(x) ∨ Food(x)
12) ¬Classic(x) ∨ Music(x)
13) ¬Jazz(x) ∨ Music(x)
14) ¬Rock(x) ∨ Music(x)
15) [¬Student(x) ∨ ¬Speaks(x, French) ∨ (Like(x, Jazz)] ∧ [¬Student(x) ∨ ¬Speaks(x, French) ∨ ¬Like(x, Rock)]
16) ¬Student(x) ∨ ¬Speaks(x, Russian) ∨ (Like(x, Rock))
17) [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ Speak(x,English)] ∧ [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ ¬Speak(x,Turkish)]
18) ¬Student(x) ∨ ¬Like(x, Fish) ∨ ¬Like(x, Classic) ∨ (Speak(x, Turkish))
19) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock)
20) ¬Music(x) ∨¬ Like(Arda,x) ∨ (¬Like(Cihan, x))
21) ¬Music(x) ∨ Like(Arda,x) ∨ (Like(Cihan, x))
22) Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)
23) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)
24) ¬α: ¬Speak(Arda, French)
25) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock) ∧ Student(Arda) by **using 19 and 5** (x: Arda)
26) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic)] ∧ Student(Arda) ∧ ¬Speaks(Arda,Russian) by **using 26 and 16** (x: Arda)
27) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic)] ∧ **¬Speaks(Arda,Russian) ∧ [Speak(Arda,English)] ∧ [¬Speak(Arda,Turkish)] by using 26 and 17** (x: Arda)
28) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic)] ∧ **¬Speaks(Arda,Russian) ∧ [Speak(Arda,English)] ∧ [¬Speak(Arda,Turkish)] ∧ ¬Speak(Arda, French) by using 27 and 24** (x: Arda)
29) {} by using 28 and 9 (x: Arda) → KB ∧ ¬α is unsatisfiable therefore α is True (Arda speaks French)

Using Proof by Contradiction for query Speak(Cihan, French)

1) ¬English(x) ∨ Language(x)
2) ¬French(x) ∨ Language(x)
3) ¬Russian(x) ∨ Language(x)
4) ¬Turkish(x) ∨ Language(x)
5) Student(Arda)
6) Student(Cihan)
7) Student(Gamze)
8) University(G(x)) ∧ Enrolled(Arda,G(x)) ∧ Enrolled(Cihan,G(x)) ∧ Enrolled(Gamze,G(x))
9) ¬Student(x) ∨ ¬University(x) ∨ (Speak(x, Language(G_1(x))) ∧ Speak(x, Language(G_2(x))) ∧ (G_1(x)≠G_2(x)))
10) ¬Fish(x) ∨ Food(x)
11) ¬Hamburger(x) ∨ Food(x)
12) ¬Classic(x) ∨ Music(x)
13) ¬Jazz(x) ∨ Music(x)
14) ¬Rock(x) ∨ Music(x)
15) [¬Student(x) ∨ ¬Speaks(x, French) ∨ (Like(x, Jazz)] ∧ [¬Student(x) ∨ ¬Speaks(x, French) ∨ ¬Like(x, Rock)]
16) ¬Student(x) ∨ ¬Speaks(x, Russian) ∨ (Like(x, Rock))
17) [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ Speak(x,English)] ∧ [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ ¬Speak(x,Turkish)]
18) ¬Student(x) ∨ ¬Like(x, Fish) ∨ ¬Like(x, Classic) ∨ (Speak(x, Turkish))
19) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock)
20) ¬Music(x) ∨¬ Like(Arda,x) ∨ (¬Like(Cihan, x)
21) ¬Music(x) ∨ Like(Arda,x) ∨ (Like(Cihan, x))
22) Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)
23) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)
24) ¬α: ¬Speak(Cihan, French)
25) [Like(Arda,Rock) ∨ (Like(Cihan, Rock))] ∧ Student(Cihan) **by combining 6 and 21** (x: Rock)
26) [Like(Cihan,Rock)] ∧ Student(Cihan) by combining 25 and 19
27)[¬Speaks(Cihan, French) ∨ (Like(Cihan, Jazz)] ∧ [**¬Speaks(Cihan,French)**] ∧ [Like(Cihan,Rock)] ∧ Student(Cihan) **by combining 26 and 15 (x: Cihan) → we obtained our query ¬Speaks(Cihan, French) by combining other expressions in our knowledge base therefore the query is false (Cihan does not speak French)**

Using Proof by Contradiction for query Speak(Gamze, French)

1) ¬English(x) ∨ Language(x)
2) ¬French(x) ∨ Language(x)
3) ¬Russian(x) ∨ Language(x)
4) ¬Turkish(x) ∨ Language(x)
5) Student(Arda)
6) Student(Cihan)
7) Student(Gamze)
8) University(G(x)) ∧ Enrolled(Arda,G(x)) ∧ Enrolled(Cihan,G(x)) ∧ Enrolled(Gamze,G(x))
9) ¬Student(x) ∨ ¬University(x) ∨ (Speak(x, Language(G_1(x))) ∧ Speak(x, Language(G_2(x))) ∧ (G_1(x)≠G_2(x)))
10) ¬Fish(x) ∨ Food(x)
11) ¬Hamburger(x) ∨ Food(x)
12) ¬Classic(x) ∨ Music(x)
13) ¬Jazz(x) ∨ Music(x)
14) ¬Rock(x) ∨ Music(x)
15) [¬Student(x) ∨ ¬Speaks(x, French) ∨ (Like(x, Jazz)] ∧ [¬Student(x) ∨ ¬Speaks(x, French) ∨ ¬Like(x, Rock)]
16) ¬Student(x) ∨ ¬Speaks(x, Russian) ∨ (Like(x, Rock))
17) [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ Speak(x,English)] ∧ [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ ¬Speak(x,Turkish)]
18) ¬Student(x) ∨ ¬Like(x, Fish) ∨ ¬Like(x, Classic) ∨ (Speak(x, Turkish))
19) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock)
20) ¬Music(x) ∨¬ Like(Arda,x) ∨ (¬Like(Cihan, x))
21) ¬Music(x) ∨ Like(Arda,x) ∨ (Like(Cihan, x))
22) Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)
23) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)
24) ¬α: ¬Speak(Gamze, French)
25) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock) ∧ Student(Gamze) **by combining 23 and 7**
26) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock) ∧ Student(Gamze) ∧ [**¬Speaks(Gamze, French)**] ∧ [¬Speaks(Gamze, French) ∨ ¬Like(Gamze, Rock)] **by combining 25 and 15 (x: Gamze) → We obtained our contradicted query ¬Speak(Gamze, French) by combining other expressions in our knowledge base. Therefore Speaks(Gamze, French) is False (Gamze does not speak French)**

Results: ∀x (Student(x) ∧ Speak(x, French)) ≡ Which student(s) speak(s) French ≡ {Arda}

2) Which student(s) speak(s) both English and Turkish

With the resolution algorithm we can only determine whether a query is True or False. Therefore for this query, (which expects a group of students as an answer) we have to determine whether each student can speak both English and Turkish seperately.

Students = {Arda, Cihan, Gamze}

∀x (Student(x) ∧ Speak(x, English) ∧ Speak(x, Turkish)) Which contains the queries:

Speak(Arda, Turkish) ∧ Speak(Arda, English), Speak(Cihan, Turkish) ∧ Speak(Cihan, English), Speak(Gamze, Turkish) ∧ Speak(Gamze, English)

Using Proof by Contradiction for query: Speak(Cihan, English) ∧ Speak(Cihan, Turkish)

1) ¬English(x) ∨ Language(x)
2) ¬French(x) ∨ Language(x)
3) ¬Russian(x) ∨ Language(x)
4) ¬Turkish(x) ∨ Language(x)
5) Student(Arda)
6) Student(Cihan)
7) Student(Gamze)
8) University(G(x)) ∧ Enrolled(Arda,G(x)) ∧ Enrolled(Cihan,G(x)) ∧ Enrolled(Gamze,G(x))
9) ¬Student(x) ∨ ¬University(x) ∨ (Speak(x, Language(G_1(x))) ∧ Speak(x, Language(G_2(x))) ∧ (G_1(x)≠G_2(x)))
10) ¬Fish(x) ∨ Food(x)
11) ¬Hamburger(x) ∨ Food(x)
12) ¬Classic(x) ∨ Music(x)
13) ¬Jazz(x) ∨ Music(x)
14) ¬Rock(x) ∨ Music(x)
15) [¬Student(x) ∨ ¬Speaks(x, French) ∨ (Like(x, Jazz)] ∧ [¬Student(x) ∨ ¬Speaks(x, French) ∨ ¬Like(x, Rock)]
16) ¬Student(x) ∨ ¬Speaks(x, Russian) ∨ (Like(x, Rock))
17) [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ Speak(x,English)] ∧ [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ ¬Speak(x,Turkish)]
18) ¬Student(x) ∨ ¬Like(x, Fish) ∨ ¬Like(x, Classic) ∨ (Speak(x, Turkish))
19) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock)
20) ¬Music(x) ∨¬ Like(Arda,x) ∨ (¬Like(Cihan, x))
21) ¬Music(x) ∨ Like(Arda,x) ∨ (Like(Cihan, x))
22) Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)
23) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)
24) ¬α: ¬Speak(Cihan, Turkish) ∨ ¬Speak(Cihan, English)
25) Student(Cihan) ∧ Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish) **by combining 6 and 22**
26) [Speak(Cihan,English)] ∧ [¬Speak(Cihan,Turkish)] ∧ Student(Cihan) ∧ Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish) **by combining 25 and 17 (x:Cihan)**
27) [¬Speak(Cihan, Turkish)] ∧ ¬[Speak(Cihan, English)] ∧ **[¬Speak(Cihan,English) ∨ ¬Speak(Cihan,Turkish)]** ∧ Student(Cihan) ∧

Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish) **by combining 26 and 24 → we obtained our query ¬α by combining other expressions. Therefore α is False (Cihan does not speak both English and Turkish)**

Using Proof by Contradiction for query: Speak(Arda, English) ∧ Speak(Arda, Turkish)

1) ¬English(x) ∨ Language(x)
2) ¬French(x) ∨ Language(x)
3) ¬Russian(x) ∨ Language(x)
4) ¬Turkish(x) ∨ Language(x)
5) Student(Arda)
6) Student(Cihan)
7) Student(Gamze)
8) University(G(x)) ∧ Enrolled(Arda,G(x)) ∧ Enrolled(Cihan,G(x)) ∧ Enrolled(Gamze,G(x))
9) ¬Student(x) ∨ ¬University(x) ∨ (Speak(x, Language(G_1(x))) ∧ Speak(x, Language(G_2(x))) ∧ (G_1(x)≠G_2(x)))
10) ¬Fish(x) ∨ Food(x)
11) ¬Hamburger(x) ∨ Food(x)
12) ¬Classic(x) ∨ Music(x)
13) ¬Jazz(x) ∨ Music(x)
14) ¬Rock(x) ∨ Music(x)
15) [¬Student(x) ∨ ¬Speaks(x, French) ∨ (Like(x, Jazz)] ∧ [¬Student(x) ∨ ¬Speaks(x, French) ∨ ¬Like(x, Rock)]
16) ¬Student(x) ∨ ¬Speaks(x, Russian) ∨ (Like(x, Rock))
17) [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ Speak(x,English)] ∧ [¬Student(x) ∨ ¬Like(x, Hamburger) ∨ ¬Speak(x,Turkish)]
18) ¬Student(x) ∨ ¬Like(x, Fish) ∨ ¬Like(x, Classic) ∨ (Speak(x, Turkish))
19) Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock)
20) ¬Music(x) ∨¬ Like(Arda,x) ∨ (¬Like(Cihan, x))
21) ¬Music(x) ∨ Like(Arda,x) ∨ (Like(Cihan, x))
22) Like(Cihan, Hamburger) ∧ ¬Like(Cihan,Fish)
23) Like(Gamze, Fish) ∧ Like(Gamze, Classic) ∧ ¬Like(Gamze, Hamburger) ∧ ¬Like(Gamze, Jazz) ∧ ¬Like(Gamze, Rock)
24) ¬α: ¬Speak(Arda, Turkish) ∨ ¬Speak(Arda, English)
25) Student(Arda) ∧ Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock) **by combining 5 and 19**
26) [Speak(Arda,English)] ∧ [¬Speak(Arda,Turkish)] ∧ Student(Arda) ∧ Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock) **by combining 25 and 17 (x:Arda)**
27) [¬Speak(Arda, Turkish)] ∧ ¬[Speak(Arda, English)] ∧ **[¬Speak(Arda,English) ∨ ¬Speak(Arda,Turkish)]** ∧ Student(Arda) ∧ Like(Arda, Jazz) ∧ Like(Arda, Hamburger) ∧ Like(Arda, Fish) ∧ ¬Like(Arda,Classic) ∧ ¬Like(Arda, Rock) **by combining 26 and 24 → we obtained our query ¬α by combining other expressions. Therefore α is False (Arda does not speak both English and Turkish)**

Problem 3)

Here I implemented the Minimax algorithm with and without using alpha-beta pruning.

To run the program, enter the code directory and run the command:
python ./main.py <minimax or alphabeta> <filename>
to run the program.

The class (node) used, which is called DotsAndBoxes, is located in the folder dots_and_boxes.py.
The minimax without pruning algorithm is located in the folder dots_and_boxes_ai.py with the function name minimax().
The minimax with alpha beta pruning algorithm is located in the folder dots_and_boxes_ai.py with the function name alphabeta().

The runtime of the 1x1 and 1x2 boards finish almost instantly when using both methods.
The runtime of the 2x2 board takes 8-10 seconds to finish using alpha-beta pruning method.
The runtime of the 2x2 board takes over 15 minutes (I did not wait for it to finish) to finish using minimax without pruning method.