Middle East Technical University Department of Computer Engineering

CENG 242

Programming Language Concepts

Spring '2011-2012

Programming Assignment 6

Due date: 9 June 2012, Saturday, 23:55

1 Introduction

In the United States, War Department in Washington, DC, General George Marshall is informed that three of four brothers in the Ryan family have all died within days of each other. Their mother will receive all three telegrams on the same day. He learns that the fourth son, James Francis Ryan is somewhere in Normandy. Then, he orders that Ryan to be found. So, he charged Capt. John H. Miller and his team to find Private Ryan there. After Miller's team arrives Normandy, they learn the place of Private Ryan. Their aim is to find Private Ryan and bring him to the safe area. They are now in the **main area**. Normandy consists different areas and Ryan is in one of the areas. Areas are connected by secure places to reach the another areas. There are also barriers in some areas. The team is able to jump over the barriers. When they find Ryan, they see that Private Ryan is wounded. He cannot walk and they have to carry him to the **safe area**. Since they are carrying Private Ryan, they will not be able to jump over the barriers anymore. They have to change their way, when they encounter barriers to reach safe area.

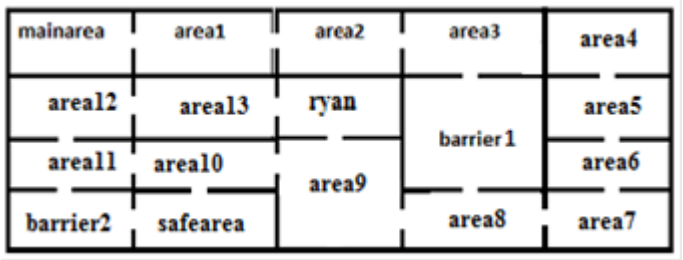
In this homework your aim is to write the Prolog code to find all the possible paths to be followed to find Private Ryan and bring him to the safe area. You will write a predicate called **save_ryan/1**.

2 Specifications

An example of the map of Normandy and prolog code for this map is shown below. Two predicates will be used to to show dependecies between areas. First of them is, **adjacent_area/2** which shows two adjacent areas. It takes two adjacent areas as parameters. The second one is **barriers/1** that shows list of barriers which are in the map. It takes list of barriers as parameter. You will write your code according to these predicates. You can create a prolog file(for example 'map1.pl') and copy them into it. Then, use **consult('map1.pl')**. command to use the map in your program. You should be careful about names. **mainarea, ryan, safearea, barriers and adjacent_area** are the names of atoms and predicates that must be written correctly in your program. Your code will be tested on different maps. **You can also create your own maps and test your program on them.**

```
adjacent_area(mainarea,area1).
adjacent_area(area2,area3).
adjacent_area(barrier1,area8).
adjacent_area(area4,area5).
adjacent_area(area6,area7).
adjacent_area(area8,area9).
adjacent_area(area9,area10).
adjacent_area(area11,area12).
adjacent_area(area13,ryan).
adjacent_area(barrier2,safearea).
barriers([barrier1,barrier2]).
```

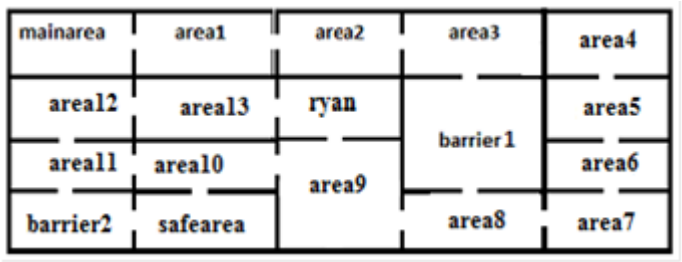
```
adjacent_area(area1,area2).
adjacent_area(area3,barrier1).
adjacent_area(area3,area4).
adjacent_area(area5,area6).
adjacent_area(area7,area8).
adjacent_area(area9,ryan).
adjacent_area(area10,area11).
adjacent_area(area12,area13).
adjacent_area(area11,barrier2).
adjacent_area(area10,safearea).
```



While the team is trying to find Ryan, they should not pass the same area more than once. After Ryan is founded, they will head to safe area. While they are trying to reach safe area, again, they should not pass the same area more than once. **mainarea,ryan and safearea** should be appear only once. Namely, the path will be in this form; [**mainarea**, distinctareas1, **ryan**, distinctareas2, **safearea**]
distinctareas1 and **distinctareas2** contains distinct areas separately. For example, area1 may appear twice on the path. However, in this case, one of them should be in distinctareas1 and the other one is in distinctareas2. Namely, an area cannot be appear twice neither in **distinctareas1** nor **distinctareas2**.

3 Sample Input and Output

Sample I



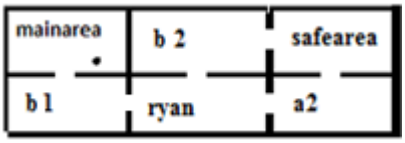
map1.pl

```
adjacent_area(mainarea,area1).          adjacent_area(area1,area2).
adjacent_area(area2,area3).             adjacent_area(area3,barrier1).
adjacent_area(barrier1,area8).           adjacent_area(area3,area4).
adjacent_area(area4,area5).              adjacent_area(area5,area6).
adjacent_area(area6,area7).              adjacent_area(area7,area8).
adjacent_area(area8,area9).              adjacent_area(area9,ryan).
adjacent_area(area9,area10).             adjacent_area(area10,area11).
adjacent_area(area11,area12).            adjacent_area(area12,area13).
adjacent_area(area13,ryan).              adjacent_area(area11,barrier2).
adjacent_area(barrier2,safearea).        adjacent_area(area10,safearea).
barriers([barrier1,barrier2]).
```

Output

save_ryan(X).
X=[mainarea,area1,area2,area3,barrier1,area8,area9,ryan,area13,area12,area11,area10,safearea] ? ;
X=[mainarea,area1,area2,area3,barrier1,area8,area9,ryan,area9,area10,safearea] ? ;
X=[mainarea,area1,area2,area3,area4,area5,area6,area7,area8,area9,ryan,area13,area12,area11,area10,safearea] ? ;
X=[mainarea,area1,area2,area3,area4,area5,area6,area7,area8,area9,ryan,area9,area10,safearea] ? ;
X=[mainarea,area1,area2,area3,barrier1,area8,area9,area10,area11,area12,area13,ryan,area13,area12,area11,area10,safearea] ? ;
X=[mainarea,area1,area2,area3,barrier1,area8,area9,area10,area11,area12,area13,ryan,area9,area10,safearea] ? ;
X=[mainarea,area1,area2,area3,area4,area5,area6,area7,area8,area9,area10,area11,area12,area13,ryan,area13,area12,area11,area10,safearea] ? ;
X=[mainarea,area1,area2,area3,area4,area5,area6,area7,area8,area9,area10,area11,area12,area13,ryan,area9,area10,safearea] ?

Sample II

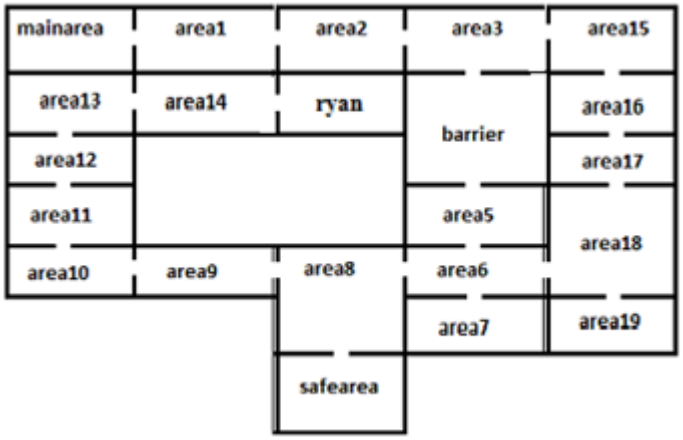


map2.pl

```
adjacent_area(mainarea,b1). adjacent_area(b1,ryan).
adjacent_area(ryan,b2). adjacent_area(ryan,a2).
adjacent_area(b2,safearea). adjacent_area(a2,safearea).
barriers([b1,b2]).
```

Output
save_ryan(X).
X = [mainarea, b1, ryan, a2, safearea]

Sample III



map3.pl

```
adjacent_area(mainarea,area1).  adjacent_area(area1,area2).
adjacent_area(area2,area3).     adjacent_area(area3,barrier1).
adjacent_area(area3,area15).    adjacent_area(area5,area6).
adjacent_area(area6,area7).     adjacent_area(area6,area8).
adjacent_area(area8,area9).     adjacent_area(area8,safearea).
adjacent_area(area9,area10).    adjacent_area(area10,area11).
adjacent_area(area11,area12).   adjacent_area(area12,area13).
adjacent_area(area13,area14).   adjacent_area(area14,ryan).
adjacent_area(area15,area16).   adjacent_area(barrier1,area5).
adjacent_area(area16,area17).   adjacent_area(area17,area18).
adjacent_area(area18,area19).   adjacent_area(area18,area6).
barriers([barrier1]).
```

Output
save_ryan(X).
X=[mainarea,area1,area2,area3,barrier1,area5,area6,area8,area9,area10,area11,area12,area13,area14,ryan,area14,area13,area12,area11,area10,area9,area8,safearea] ? ;
X=[mainarea,area1,area2,area3,area15,area16,area17,area18,area6,area8,area9,area10,area11,area12,area13,area14,ryan,area14,area13,area12,area11,area10,area9,area8,safearea] ?

4 Regulations

1. **Programming Language:** You should write your code using GNU Prolog and test them on inek machines
2. **Late Submission:** See the syllabus
3. **Cheating:** All the work should be done individually. **We have zero tolerance policy for cheating**. Students involved cheating will get 0 from all homeworks and they will be punished according to the university regulations.
4. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.

5 Submission

1. Submit a single file called "hw6.pl" through the COW system.
2. The evaluation will be black box. However, we will check your code for cheating.