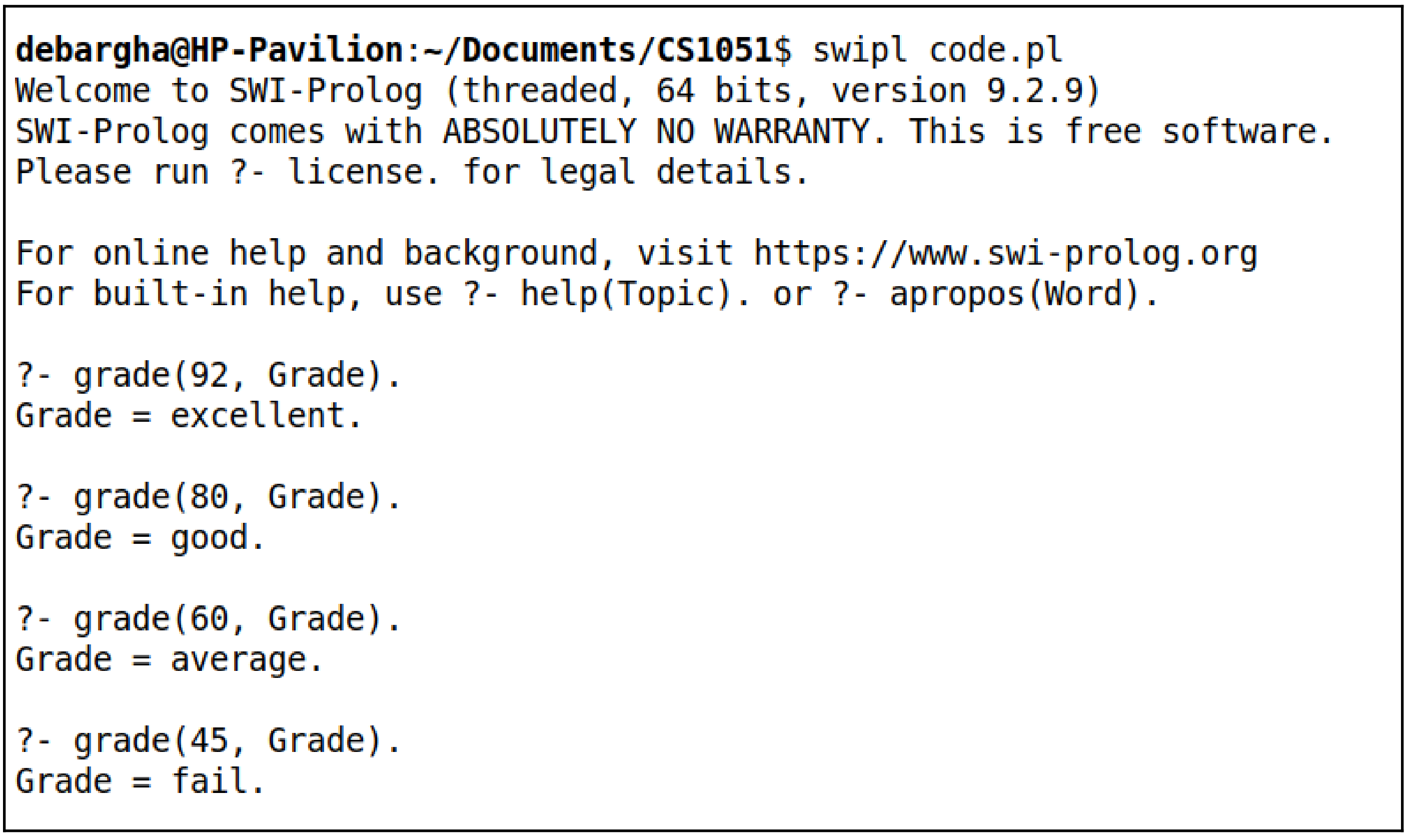
Assignment-3

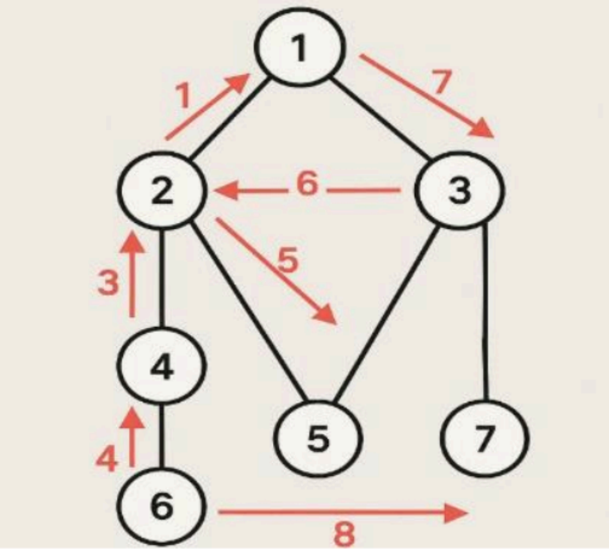
1. **Write a Prolog program that takes a student's marks as input and returns their grade according to the following rules:**
   - **Marks >= 90 → grade = excellent**
   - **Marks >= 75 and < 90 → grade = good**
   - **Marks >= 50 and < 75 → grade = average**
   - **Marks < 50 → grade = fail**

code.pl

```prolog
grade(Marks, excellent) :-
    Marks >= 90, !.
grade(Marks, good) :-
    Marks >= 75,
    Marks < 90, !.
grade(Marks, average) :-
    Marks >= 50,
    Marks < 75, !.
grade(Marks, fail) :-
    Marks < 50.
```



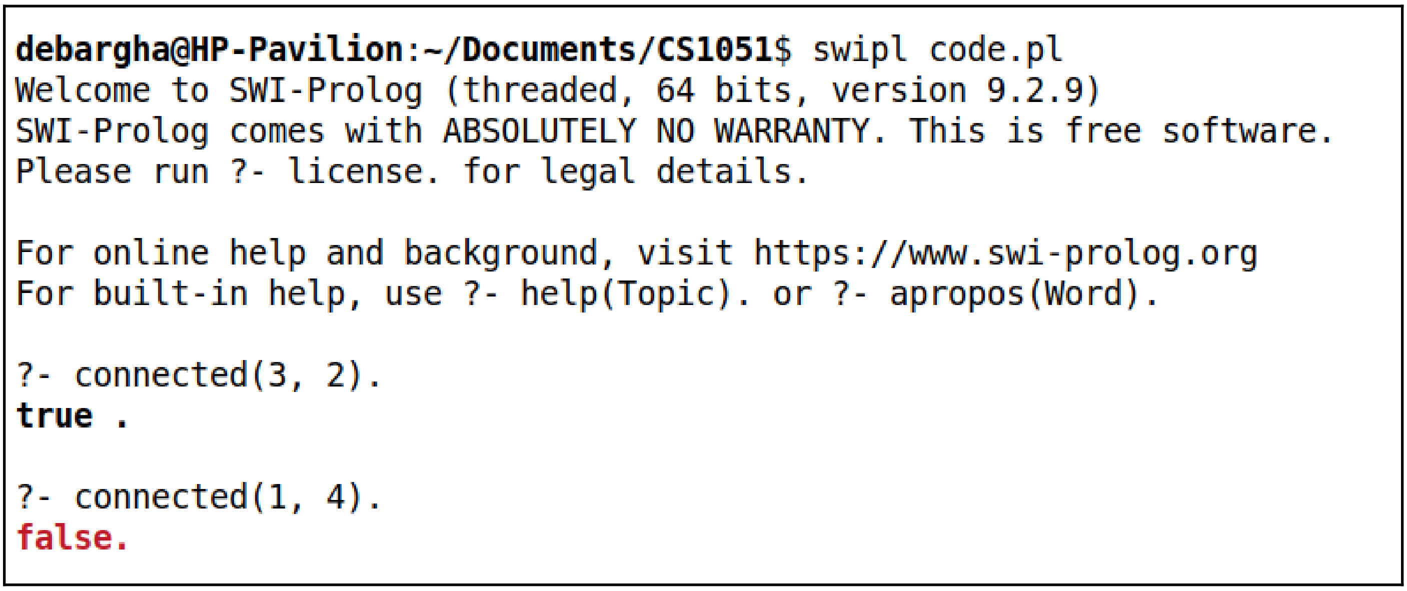
```
debargha@HP-Pavilion:~/Documents/CS1051$ swipl code.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- grade(92, Grade).
Grade = excellent.

?- grade(80, Grade).
Grade = good.

?- grade(60, Grade).
Grade = average.

?- grade(45, Grade).
Grade = fail.
```



2. **Store all edges of Graph A and check if any path exists or not.**

code.pl

```
edge(1, 3).
edge(2, 1).
edge(3, 2).
edge(2, 5).
edge(3, 5).
edge(3, 7).
edge(4, 6).
edge(2, 4).
connected(X, Y):-
      edge(X, Y).
connected(X, Y):-
      edge(Y, X).
```

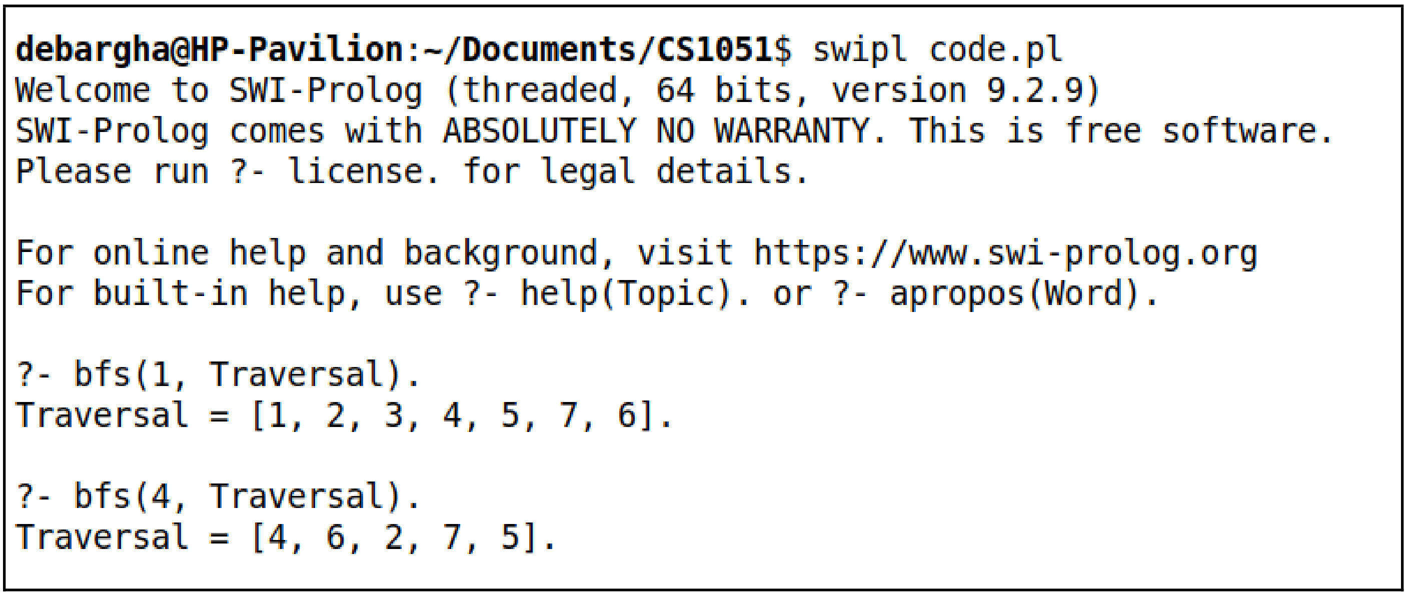


```
debargha@HP-Pavilion:~/Documents/CS1051$ swipl code.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- connected(3, 2).
true .

?- connected(1, 4).
false.
```

3. Implement Breadth-First Search (BFS) traversal of Graph A in Prolog.

code.pl

```prolog
edge(1, 2).
edge(1, 3).
edge(2, 4).
edge(4, 6).
edge(6, 2).
edge(2, 5).
edge(3, 2).
edge(3, 5).
edge(3, 7).
edge(6, 7).

bfs(Start, Traversal) :-
    bfs_queue([Start], [], Traversal).

bfs_queue([], Visited, Traversal) :-
    reverse(Visited, Traversal).

bfs_queue([Node|RestQueue], Visited, Traversal) :-
    member(Node, Visited), !,
    bfs_queue(RestQueue, Visited, Traversal).

bfs_queue([Node|RestQueue], Visited, Traversal) :-
    findall(Next, edge(Node, Next), Neighbors),
    append(RestQueue, Neighbors, NewQueue),
    bfs_queue(NewQueue, [Node|Visited], Traversal).
```

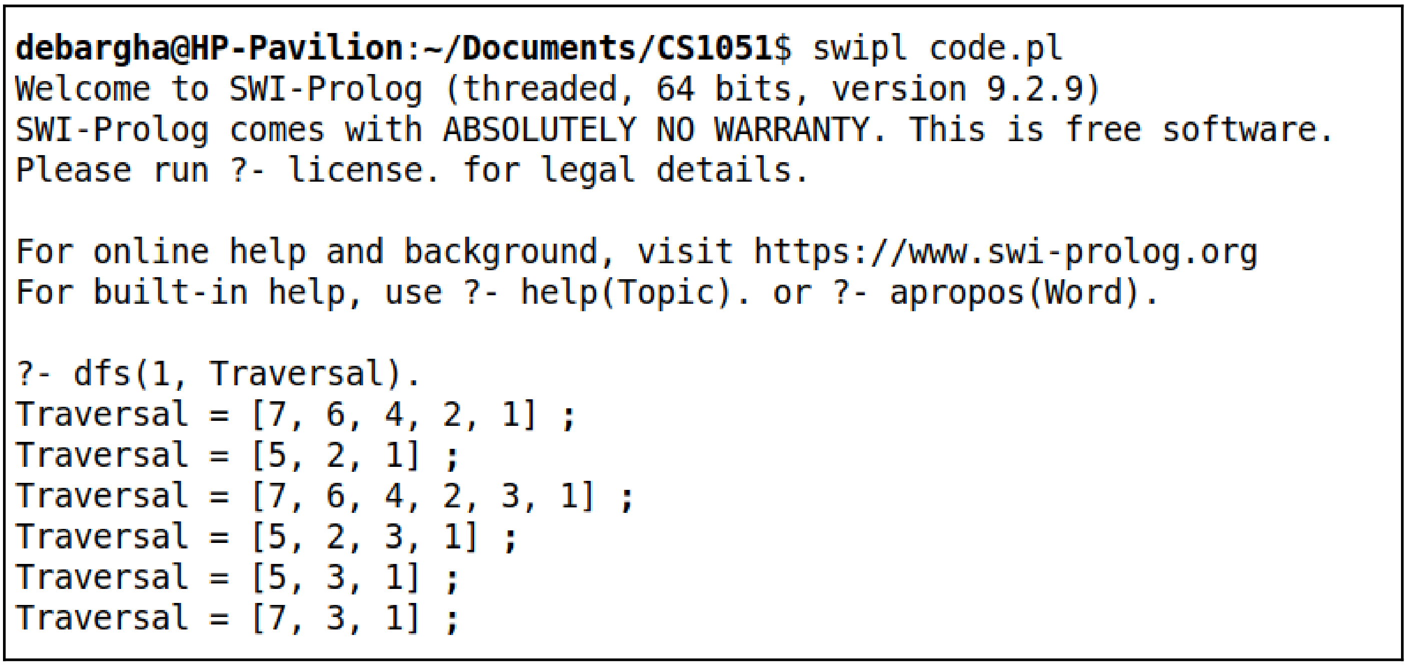


```
debargha@HP-Pavilion:~/Documents/CS1051$ swipl code.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- bfs(1, Traversal).
Traversal = [1, 2, 3, 4, 5, 7, 6].

?- bfs(4, Traversal).
Traversal = [4, 6, 2, 7, 5].
```

4. Implement Depth-First Search (DFS) traversal of Graph A in Prolog.

code.pl

```prolog
edge(1, 2).
edge(1, 3).
edge(2, 4).
edge(4, 6).
edge(6, 2).
edge(2, 5).
edge(3, 2).
edge(3, 5).
edge(3, 7).
edge(6, 7).

dfs(Start, Traversal) :-
    dfs_helper(Start, [], Traversal).

dfs_helper(Node, Visited, [Node|Visited]) :-
    findall(Next, edge(Node, Next), Neighbors),
    \+ (member(NextNode, Neighbors), \+ member(NextNode, Visited)).

dfs_helper(Node, Visited, Traversal) :-
    findall(Next, edge(Node, Next), Neighbors),
    member(NextNode, Neighbors),
    \+ member(NextNode, Visited),
    dfs_helper(NextNode, [Node|Visited], Traversal).
```



```
debargha@HP-Pavilion:~/Documents/CS1051$ swipl code.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- dfs(1, Traversal).
Traversal = [7, 6, 4, 2, 1] ;
Traversal = [5, 2, 1] ;
Traversal = [7, 6, 4, 2, 3, 1] ;
Traversal = [5, 2, 3, 1] ;
Traversal = [5, 3, 1] ;
Traversal = [7, 3, 1] ;
```