BSC DEGREE IN COMPUTER SCIENCE / COMPUTER ENGINEERING /

AND SOFTWARE ENGINEERING

**Artificial Intelligence (practical Exam)**

Name : MH Mihiranga

Reg No : D-BCE-22-0014

01).

A white rectangular object with a black border

Description automatically generated

02).

A screen shot of a computer

Description automatically generated

**OUTPUT**

**A close-up of a code

Description automatically generated**

**A close-up of a computer code

Description automatically generated**

03).A group of text on a white background

Description automatically generated

A screen shot of a computer code

Description automatically generated

**04) .**

**A screenshot of a computer code

Description automatically generated**

**A close-up of a computer code

Description automatically generated**

**A close-up of a paper

Description automatically generated**

**Full Code**

student('Hasitha',101,'Gampaha').

student('Ramith',102,'Kaluthara').

student('Tharidu',103,'Dankotuva').

student('Amal',104,'Mathara').

result('101','CS2000',66).

result('102','CE2000',78).

result('103','CD2890',67).

result('104','CV1234',45).

course('CS2000','DBMS',2).

course('CE2000','AI',3).

course('CD2890','GPHW',4).

course('CV1234','OOP',3).

addstudent:-

write('Enter student name : '),read(name),

write('Enter Index no : '),read(id),

write('Enter address : '),read(add),

assert(student(name,id,add)),

write('New Student added successfully.').

:-dynamic student/3.

addResult:-

write('Enter Index No: '), read(IndexNo),

write('Enter Course Code: '), read(CourseCode),

write('Enter Marks: '), read(Marks),

% Check if the student exists

student(Name, IndexNo, \_),

% Check if the course exists

course(CourseCode, \_, \_),

assert(result(IndexNo, CourseCode, Marks)),

write('Result added successfully for student '), write(Name).

addcourse:-

write('Enter course code:'),read(CourseCode),

write('Enter course name:'),read(Name),

write('Enter credits:'),read(Credit),

assert(course(CourseCode,Name,Credit)),

write('Course added successfully.').

:-dynamic course/3.

grade(Marks, 'A') :- Marks >= 70.

grade(Marks, 'B') :- Marks >= 65, Marks < 70.

grade(Marks, 'C') :- Marks >= 55, Marks < 65.

grade(Marks, 'S') :- Marks >= 40, Marks < 55.

grade(Marks, 'F') :- Marks < 40.

printStudentGrade :-

write('Enter Index No: '), read(IndexNo),

write('Enter Course Code: '), read(CourseCode),

% Check if the student exists

student(Name, IndexNo, \_),

% Check if the course exists

course(CourseCode, \_, \_),

% Check if the result exists

result(IndexNo, CourseCode, Marks),

grade(Marks, Grade),

write('Student: '), write(Name), nl,

write('Course: '), write(CourseCode), nl,

write('Marks: '), write(Marks), nl,

write('Your Grade is : '), write(Grade).

printResultSheet :-

write('Enter Index No: '), read(IndexNo),

student(Name, IndexNo, \_),

write('Student name: '), write(Name), nl,

write('Course Code Marks'), nl,

write('--------------------------------'), nl,

findall(CourseCode-Marks, result(IndexNo, CourseCode, Marks), Results),

printResults(Results).

printResults([]).

printResults([CourseCode-Marks | Rest]) :-

course(CourseCode, \_, \_),

grade(Marks, Grade),

format('~w ~t~30| ~w~n', [CourseCode, Marks]),

printResults(Rest).