Code Explanation :

--

```
   1.
:-dynamic(incompletePreRequisites/1), dynamic(courseDone/1),
dynamic(courseChecked/2), dynamic(doneNow/1), dynamic(career/1).

Explanation :
:-dynamic(<some name>/X).
This makes the clause with header name <some name> taking in X number of
arguments to be a dynamic predicate.
By default suppose for some clause, say:
hello(3).
Now for the query :
?-hello(3).
Will evaluate to true.
but
?-hello(1).
Will give an error as the clause
hello(1).
Is not defined.
But after making hello(_) a dynamic predicate, the query
?-hello(1).
Will evaluate to false.

I am making the predicates that I am using in the if-then-else construct
condition part, to be dynamic so that if those predicates are not found
then the flow goes to the false part of the if-then-else construct.
Eg:
hello(X)->statement1;statement2.
Say there is only one hello(3) defined and no hello(1) asserted till now.
Then for X=3, the above statement will turn to :
hello(3)->statement1;statement2.
(because hello(3) is a known fact in the database and hence hello(3)
evaluates to its body: which is empty and hence becomes true,
NOTE: if hello(3) is not a fact but a rule then hello(3) will evaluate to
its body and will eventually evaluate to either true (empty body) or fail.
false and correspondingly the flow will go to statement1 or statement2)
And the final evaluation value of the entire statement is what statement1
evaluates to.
For X=1, the statement turns to :
```

hello(1)->statement1;statement2.
hello(1) is not a known fact in the database and hence
if hello/1 (1 being the number of inputs hello takes) is static then after this the statement gives ‘Unknown Procedure Error’.
But if hello/1 is dynamic then hello(1) evaluates to false, and the flow goes to statement2.
I am doing this to implement the sequential if-else like construct where the condition to be checked is the existence of a predicate.

2.

```
:-['courses.txt'].
:-['careers.txt'].
:-['preRequisites.txt'].
```

These queries just loads the prolog instruction in these text files before loading the rest of the program.
This is just like import in java.
courses.txt contains all the known courses/skills. These are in the form :
course(X).
Where X is the name of the course/skill.
Eg :
course(digital_circuits).
careers.txt contains all the known careers. These are in the form :
career(X).
It is just like courses, where X is the name of the career.
Eg :
course(computer_programmer).
preRequisites.txt contains all the known pre-requisites for any course/skill/career.
These are in the form :
preRequisite(X, Y).
Where X is the name of the course/skill/career and Y is the name of the course/skill. This implies that Y is a prerequisite for doing X.
Eg :
preRequisite(human_centred_ai, machine_learning).
This means machine_learning course is a prerequisite to doing human_centred_ai course.

preRequisite(computer_programmer, advanced_programming).

This means that to be a computer programmer a person must have completed the advanced programming course.

3.

```
start():-write('Enter the list of Careers/Courses that you are interested in (only check for registered careers; to check the list of registered careers type : career(X)): '), read(List), list_of_careers(List), outputSuggestedCourseOrJobs(UnSortedList, 1), mergeSort(Sorted, UnSortedList, 1), outputSuggestedCourseOrJobs(Sorted, 2), asker(Sorted), cleaner, !.
```

To start the run program just type:

```
?-start().
```

In the terminal and follow instructions.

Subassertions :

4. `write('Enter the list of Careers/Courses that you are interested in (only check for registered careers; to check the list of registered careers type : career(X)): ')`

   This is just an output to be given to the user.

5. `read(List)`

   This is to take a list of career/course/skills from the user as input.

6. `list_of_careers(List)`

   To understand this we must understand :

   ```
   list_of_careers([Career|Remaining]):- checkPreRequisites(Career),
   list_of_careers(Remaining).
   list_of_careers([]).
   ```

   This is just a recursive list looping construct to call the clause checkPreRequisites/1 with all the elements of the list one by one, note that the input list is known, which means Career and Remaining get bound. Career gets bound to the 1st element in the list and Remaining gets bound to the remaining list excluding the 1st element.

   list_of_careers([Career|Remaining]) just evaluates to:
   checkPreRequisites(Career), list_of_careers(Remaining)
   Where list_of_careers(Remaining) is for recursion for the rest of the list and checkPreRequisites(Career) just call checkPreRequisites/1 for the selected element Career of the input list.

   list_of_careers([])., is the base case to stop the recursion/loop when the input list becomes empty.

7. outputSuggestedCourseOrJobs(UnSortedList, 1)
   This clause just takes and removes all the suggested courses/skills/careers and forms them into the list UnSortedList. To understand this, we must understand:
   a. Whenever a course gets suggested to be output, the statement assert(suggestedCourseOrJob(Course)) will be evaluated. This implies that all the suggested courses(which are somehow obtained during the entire program execution/after the execution of list_of_careers(List) get stored in the database in the form suggestedCourseOrJob(Course), where Course is the name of the Course which is suggested to the user).
      Eg :
      suggestedCourseOrJob(digital_circuits).
      Being stored in the database as a fact implies that digital_circuits will be suggested to the user after the program execution.
   b. outputSuggestedCourseOrJobs(UnSortedList, 1):-
      listCreator(UnSortedList), write('Unsorted list of suggested courses : '), write(UnSortedList), nl.

      To understand this we must understand :
      i. listCreator(UnSortedList)
         To understand this we must understand:
         1. listCreator([Current|Remaining]):-retract(suggestedCourseOrJob(Current)), listCreator(Remaining).
            listCreator([]).

            After the execution of list_of_careers(List), all the suggested courses/skills/jobs gets stored in the database in the form:
            suggestedCourseOrJob(Course), where Course is the name of the Course suggested.

            This is again a recursive/looping construct which after all the computations at the end of the program/after execution of list_of_careers(List) will recursively retract/1 all the suggested courses stored in the form suggestedCourseOrJob(Course) and store them in the form of the list UnSortedList.

NOTE : In the start UnSortedList is unknown. (this is similar to convert_to_list([Px|Tail]) clause in the sample prolog program sir posted (prg4.pdf))

listCreator([Current|Remaining]):-retract(suggestedCourseOrJob(Current)), listCreator(Remaining).

As the input is unknown/unbound :
Current is unbound, Remaining is unbound.
After retract(suggestedCourseOrJob(Current)), a suggested course is retracted/1 (selected and REMOVED from the database). This retracted/1 course is bound to Current. Now only Remaining is unbound which is bound using recursion by the statement : listCreator(Remaining).

listCreator([])., is evaluated when retract(suggestedCourseOrJob(Current)) evaluated to false (implying that no more suggested courses are left to be retracted/1). This makes its corresponding listCreator([Current|Remaining]) evaluated to false/fail. This leads to backtracking to the next similar clause i.e. listCreator([]). Hence, this list gets bound, now in the parent call, this makes its Remaining bind and this recursively forms the list.

2. write('Unsorted list of suggested courses : '), write(UnSortedList), nl.

   This just outputs the UnSortedList obtained in listCreator(UnSortedList), in the form :

```
Unsorted list of suggested courses :
$UnSortedList<newline>
```

In the end, we have retrieved and REMOVED all the suggested courses/skills/careers and stored them into UnSortedList.

8. mergeSort(Sorted, UnSortedList, 1)

   This is the clause for the mergeSort algorithm implemented in prolog which takes in the unsorted list UnSortedList, sorts it, and binds

Sorted with the result sorted list. As the third argument = 1. This is a form of merge sort in which all the elements of Sorted are in descending order of X, where X comes from courseChecked(Course, X), where Course is a Course in Sorted, i.e. For two courses Course1 comes before Course2 if X1>=X2, where X1 and X2 comes from courseChecked(Course1, X1) and courseChecked(Course2, X2). Further any course comes before (or is greater than) a career/job.

9. outputSuggestedCourseOrJobs(Sorted, 2)

   To understand this we must understand:
   outputSuggestedCourseOrJobs(Sorted, 2):- nl, write('The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) : '), nl, printer(Sorted, 1), !.

   a. nl, write('The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) : '), nl,
      This is a normal printing statement
   b. printer(Sorted, 1), !.
      This is a recursive/looping construct that takes in the list Sorted and prints its contents in the format specified by it recursively.
      ! just prevents backtracking (to not print again by mistake).

      To understand it we must understand:
      i. printer([Current| Remaining], Index):- write(Index), write('. '), write(Current), write(' ('), courseChecked(Current, Output), write(Output), write(')'), nl, Next is (Index+1), printer(Remaining, Next), !.
         printer([], Index).

         Here the input list Sorted is known/bound, which means Current and Remaining are bound. Index is an integer.
         1. write(Index), write('. '), write(Current), write(' ('),
            This is a normal printing statement.
         2. courseChecked(Current, Output)
            Here Current is bound, Output is unbound. Output is made available by courseChecked(_, _)

The number of courses satisfied on doing the given course(say Course)(by the student) say X is stored in the form courseChecked(Course, X) after list_of_careers(List).
courseChecked(Current, Output) just retrieves the Output corresponding to Current.
Now Output is bound.

3. `write(Output), write(')'), nl,`
   This is normal printing.
4. `Next is (Index+1)`
   Next = Index+1.
5. `printer(Remaining, Next), !`
   Recursion for the rest of the list. ! to prevent backtracking.
6. `printer([], Index).`
   For any value of Index if we get an empty input list then that is a base case.

   Index is just to increment number in front of the printing line(index value).

In the end outputSuggestedCourseOrJobs(Sorted, 2), just prints all the elements of Sorted in the order in which they appear in Sorted.

10. asker(Sorted)

```
asker(Sorted):- write('Do you want to add your interest points to
your result? (y/n) : '), nl, read(Input),
((Input=y)->(adder(Sorted), mergeSort(NewSorted, Sorted, 2), nl,
write('New changed recommendations : '), nl,
outputSuggestedCourseOrJobs(NewSorted, 2));((Input=n)->(write('ok,
best of luck')); (write('Enter a valid choice!'), nl,
asker(Sorted)))), !.
```

This is to ask and if told add interest point to X such that courseChecked(Course, X), for all suggested courses.

Here as Sorted is sorted decreasingly based on X.
We are adding interest points to each X and then again sorting the list decreasingly (using mergeSort(NewSorted, Sorted, 2)) such that

NewSorted will combine the sorting sequence/result of both Sorted and interest point addition.
Interest point: This is a number that I am obtaining from the user. This is added to X such that courseChecked(Course, X), for all suggested courses. This means that the higher the interest point, the higher the newX, where newX = X+(interest point).
! is to prevent backtracking.
This just takes an input say Input until Input is a valid choice (input=y or Input=n),
If Input is an invalid choice, then it prints an error message and again asks for input Input using recursion (see asker(Sorted)at the end of the rule. It evaluated to the same value as to what recursive call evaluated to.
If Input=n, then the rule prints something and just exits out of the asker(Sorted) (which itself evaluated to true).
If Input=y,

a. adder(Sorted)

   Here Sorted is known/bound. Which was output of mergeSort(Sorted, UnSortedList, 1).
   This is to add interest points to X such that courseChecked(Course, X), for all suggested courses.
   Understanding:
   ```
   adder([Current|Remaining]):- asker2(Current, Result),
   courseChecked(Current, Val), retractall(courseChecked(Current,
   _)), Next is (Val+Result), assert(courseChecked(Current,
   Next)), adder(Remaining), !.
   adder([]).
   ```
   ! is to prevent backtracking.
   Here Sorted is known, meaning that Current and Remaining are bound

   This is a construct to perform following computation on all elements of Sorted from left to right :

   ```
   asker2(Current, Result), courseChecked(Current, Val),
   retractall(courseChecked(Current, _)), Next is (Val+Result),
   assert(courseChecked(Current, Next))
   ```

   i. asker2(Current, Result)

      Here Current is already bound, and we get Result after its evaluation (which is initially unbound).

Here Result is the interested point obtained from the user using :

```
asker2(Course, Input):- nl, write('How much interested
are you in this Course/Job (1-5) :'), write(Course),
write(' ? : '), nl, write('1. hate it (-1 point)'), nl,
write('2. don\'t either hate or love it (+0 point)'), nl,
write('3. slightly interested in it (+1 point)'), nl,
write('4. love it (+2 points)'), nl, write('5. it is a
must do (+5 points)'), nl, read(X), ((integer(X), (X>=1,
5>=X))-> map(X, Input); nl, write('Enter a valid input :
input out of range (>=1 and <=5) or not an integer!'),
nl, asker2(Course, Input)), !.
```

Which prints :

```
How much interested are you in this Course/Job (1-5) :
Course ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
```

Which will keep on taking X (index of the output lines) until X is valid (1<=X<=5 and X is an integer), when X is valid converts the X into its corresponding interest point= Input using map(X, Input). And then Input is output from the rule.

ii. courseChecked(Current, Val), retractall(courseChecked(Current, _)), Next is (Val+Result), assert(courseChecked(Current, Next))

This retrieves the old value of Val (priority of Current in the list Sorted (which is sorted in the decreasing order of these priority points)).
Then to avoid multiple instance, before asserting, it retracts all courseChecked(Current,_) for current course int list.
Then the new priority is calculated by adding priority point to it (=Result).

Then the priority value of Current is stored as
courseChecked(Current, Next).

Essentially, adder(Sorted), just updates the priority value for all the courses in the list in left to right manner.

b. mergeSort(NewSorted, Sorted, 2)

Now as the priorities of all the courses in the list Sorted is changed, another sorting is done using
mergeSort(NewSorted, Sorted, 2)
Which sorts the result and binds the sorted list with the list NewSorted.
mergeSort(NewSorted, Sorted, 2)
Is the same as mergeSort(Sorted, UnSorted, 1), except it doesnot pushes careers/jobs to the end of the list.
Or
It just performs the sorting in the same manner, and does not enforces the condition :
"any course comes before (or is greater than) a career/job."

11. cleaner.

This is just to clean all the clauses that were asserted during the program so that the next use is not affected.
It uses :

```
cleaner:- retractall(courseDone(_)),
retractall(suggestedCourseOrJob(_)),
retractall(incompletePreRequisites(_)),
retractall(courseChecked(_,_))
```

And retracts all the asserted facts from the database.
PROGRAM ENDS AFTER CLEANER.

12.

```
checkPreRequisites(Input):- (not(courseChecked(Input,
_))->(assert(courseChecked(Input, 1)), not(branchRecurser(Input)),
suggestingCondition(Input));true).
```

This checks for all the prerequisites of the entered course/skill/job to decide whether to suggest the given course or not.
Construct is :

```
if(not(courseChecked(Input, _))){
     assert(courseChecked(Input, 1)),
     not(branchRecurser(Input)),
     suggestingCondition(Input)
```

}
To ease understanding:
Consider any course/job/skill as a directed graph node such that all the prerequisites are its children with an edge from prerequisite to the current course.

Any course/job/skill, say X will be of the following form :

(a) We know that we have done this course.

Hence we have knowledge about this X. And this course is done.

courseChecked(X, _), courseDone(X) will be in the database, where the 2nd argument of courseChecked is the number of courses whose prerequisite is X.

(b) We know that we have not done this course AND we have checked all its prerequisites AND we have checked whether all its prerequisites are done.

Hence we have knowledge about this X. And this course is not done. And this course is suggested as all its prerequisites are completed.

courseChecked(X, _), suggestedCourseOrJob(X) will be in the database, and courseDone(X) will not be in the database, where 2nd argument of courseChecked is the number of courses whose prerequisite is X.

(c)We know that we have not done this course AND we have checked all its prerequisites AND some prerequisites are not done.

Hence we have knowledge about this X. And this course is not done. And this course is not suggested as some prerequisites are pending.

courseChecked(X, _), incompletePreRequisites(X) will be in the database;suggestedCourseOrJob(X) and courseDone(X) will not be in the database, where 2nd argument of courseChecked is the number of courses whose prerequisite is X.

(d) We know that we have not done this course AND we have not checked all its prerequisites.

Hence, this course is not done and we do not have complete information about this course.

, incompletePreRequisites(X) might be in the database; courseChecked(X, _) and suggestedCourseOrJob(X) and courseDone(X) will not be in the database, where 2nd argument of courseChecked is the number of courses whose prerequisite is X.

incompletePreRequisites(X) is in the database when X has at least one non-done prerequisite (i.e. for some Course such that preRequisite(X, Course), we do not have courseDone(Course) in the database but we do have courseChecked(Course, _)).

courseDone(X) is in the database when we know that X is done by the user.

suggestedCourseOrJob(X) is in the database when all prerequisites of X are done by the user.

courseChecked(X, _) is in the database when this course is either done or when this course has been checked for prerequisites.

A. We are only looking for prerequisites for only those courses which we know nothing about.
I.e. if we enter the if block then we do not know about this course.
Now we know something about this course hence:
assert(courseChecked(Input, 1))
Now we are checking for each prerequisite using branchRecurser(Input). As we always fail in branchRecurser(Input) to enforce backtracking hence it will always evaluate to false. Hence, we use not(branchRecurser(Input)).

B. branchRecurser(Input):- preRequisite(Input, Course), isCourseDone(Input, Course), fail.

This just enforces backtracking to look for all prerequisites Course for the given course Input.

```
C. isCourseDone(Input, Course):- (not(courseChecked(Course,
   _))->(assert(doneNow(Course)), interiorCheck(Input, Course));
   ((not(courseDone(Course))-> incompletePreRequisitesSetter(Input);
   true), true)), numberOfAccessSetter(Course).

   Construct:
   if(not(courseChecked(Input, _))){
         assert(doneNow(Course)),
         interiorCheck(Input, Course)
   }
   else if(not(courseDone(Course))){
         incompletePreRequisitesSetter(Input)
   }
   numberOfAccessSetter(Course)

   I.e.
   if(this course is neither done nor checked for prerequisites){
         doneNow is acting as a flag that info about this course is
   known in this clause as was not known previously.
         Assert doneNow.
         Call interiorCheck(Input, Course)
   }
   else if(doneCourse(Course) is not present in the database){
         Call incompletePreRequisitesSetter(Input)
   }
   Call numberOfAccessSetter(Course)

D. incompletePreRequisitesSetter(Input):-(incompletePreRequisites(Input
   )-> true; assert(incompletePreRequisites(Input)))
   Construct :

   if(not(incompletePreRequisites(Input))){
         assert(incompletePreRequisites(Input))
   }
   If incompletePreRequisites(Input) was not already asserted for the
   given course then assert it.

E. interiorCheck(Input, Course):-(courseDone(Course)->
   (assert(courseChecked(Course, 1)), true) ;
(askIfCourseIsDone(Course) -> (assert(courseChecked(Course, 1)),
assert(courseDone(Course))); (incompletePreRequisitesSetter(Input),
checkPreRequisites(Course), (not(courseChecked(Course,
_))->(assert(courseChecked(Course, 1))); true)/*,
(suggestedCourseOrJob(Course)->(incompletePreRequisites(Input)->
true; assert(incompletePreRequisites(Input))); true)*/))).
```

```
Construct:
if(courseDone(Course)){
      assert(courseChecked(Course, 1))
}
else if(askIfCourseIsDone(Course)){
      assert(courseChecked(Course, 1))
      assert(courseDone(Course))
}
else {
      incompletePreRequisitesSetter(Input)
      checkPreRequisites(Course)
      if(not(courseChecked(Course, _))){
            assert(courseChecked(Course, 1))
      }
}
```

```
I.e.
It is guaranteed that in interiorCheck(Input, Course) Input will
always be different for a given course, which means that we can
increase the priority number of the Course.
It is also given that Course will always be not known in the
database i.e. courseChecked(Course, _) will not exist (check the
construct of isCourse(Course, Input))
if(Course if done){
      Now we know about the course.
      Hence assert.
}
else if(user has done the Course){
      Now we know about the course and the course is done hence the
assertions.
}
else{
```

If Input doesnot have incompletePreRequisites(Input) already in database then assert it.

Now go further down the branch of this course by recursion and check for its prerequisites.

If still, info about course is not available even after checkPreRequisites(Course) then assert it as available as now this course/node is traversed

}

F. `askIfCourseIsDone(Course):-write("Have you done this course : "), write(Course), write('? (y/n)'), nl, read(Answer), (Answer = y -> true; (Answer = n -> false; write('Enter a valid choice!'), nl, askIfCourseIsDone(Course))).`

This just keeps on asking the question for some course Course:
Have you done this course : Course? (y/n)
Until we get a valid input (= Answer) i.e. Answer = y or Answer = n.
Otherwise the code keeps asking using recursion.
If this evaluates to true then user has done this course else if it evaluates to false then user has not done this course.

G. `incompletePreRequisitesSetter(Input):-(incompletePreRequisites(Input)-> true; assert(incompletePreRequisites(Input))).`

Construct:

```
if(not(incompletePreRequisites(Input))){
      assert(incompletePreRequisites(Input))
}
```

If Input doesnot have incompletePreRequisites(Input) already in database then assert it.

H. `numberOfAccessSetter(Course):-`
`(doneNow(Course)->(retractall(doneNow(Course)));`
`(increaseNumberOfAccesses(Course)))`

Construct:

```
if(doneNow(Course)){
      retractall(doneNow(Course))
}
```

```
else {
      increaseNumberOfAccesses(Course))
}
```
If the doneNow(Course) fact is set, then retract it else increase the priority number of the Course.

I. increaseNumberOfAccesses(Course):- courseChecked(Course, Val), NewVal is (Val+1), retractall(courseChecked(Course, _)), assert(courseChecked(Course, NewVal)), !.

Construct:
For any X such that courseChecked(Course, X), it retracts courseChecked(Course, X) and asserts courseChecked(Course, X+1). Basically increasing the priority number of Course.

J. suggestingCondition(Input):- (incompletePreRequisites(Input)-> retract(incompletePreRequisites(Input)); assert(suggestedCourseOrJob(Input))).

Construct:
```
if(incompletePreRequisites(Input)){
      retract(incompletePreRequisites(Input))
}
else{
      assert(suggestedCourseOrJob(Input)))
}
```

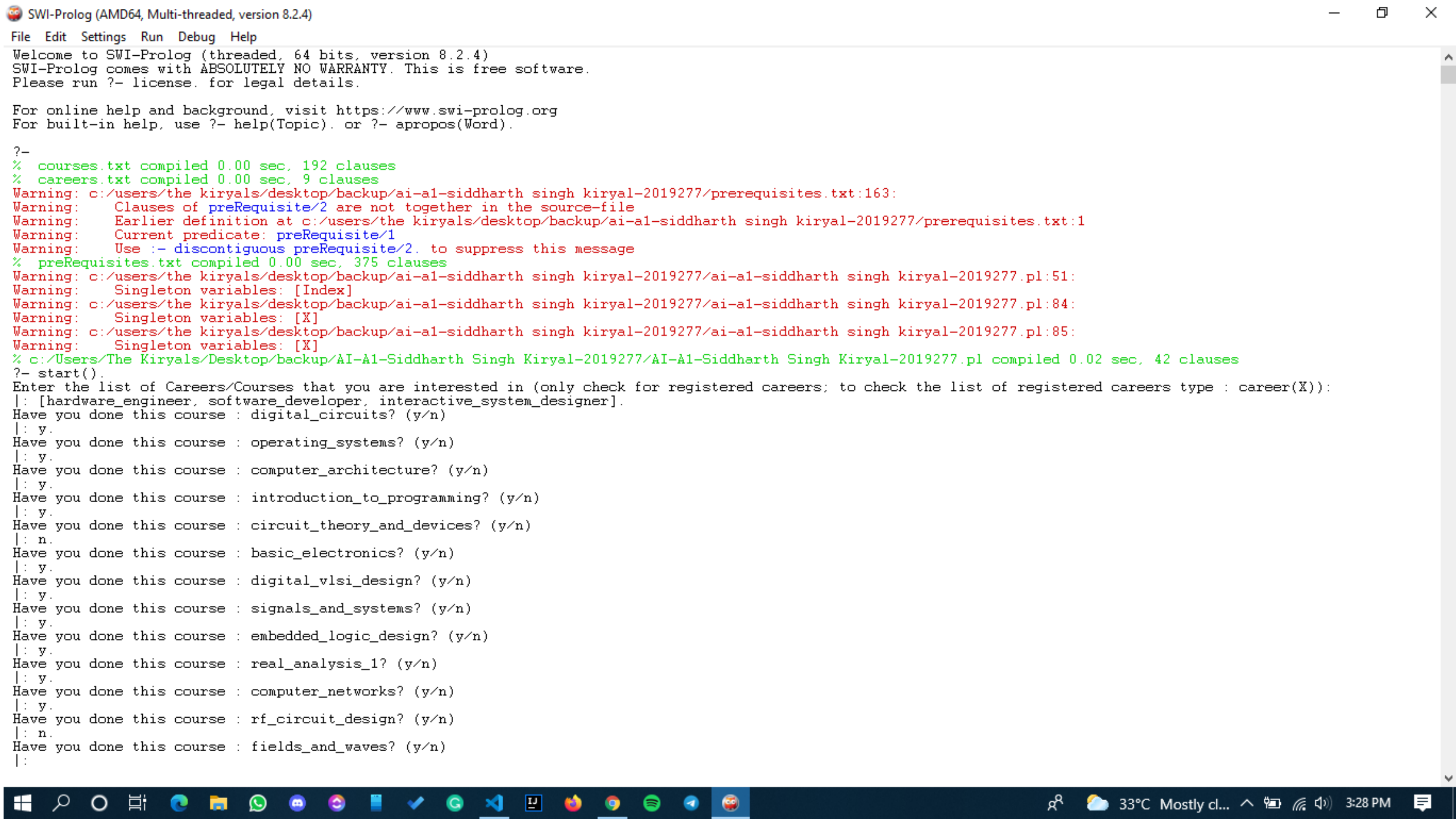
Ie.
```
if(at least one prerequisite of Input is not done and we know
everything about it){
      Remove the clause incompletePreRequisites(Input) from the
database for cleaning purposes
}
else {
      Suggest Input
}
```

13. mergeSort(Sorted, List, X) :
Takes unsorted known input list :

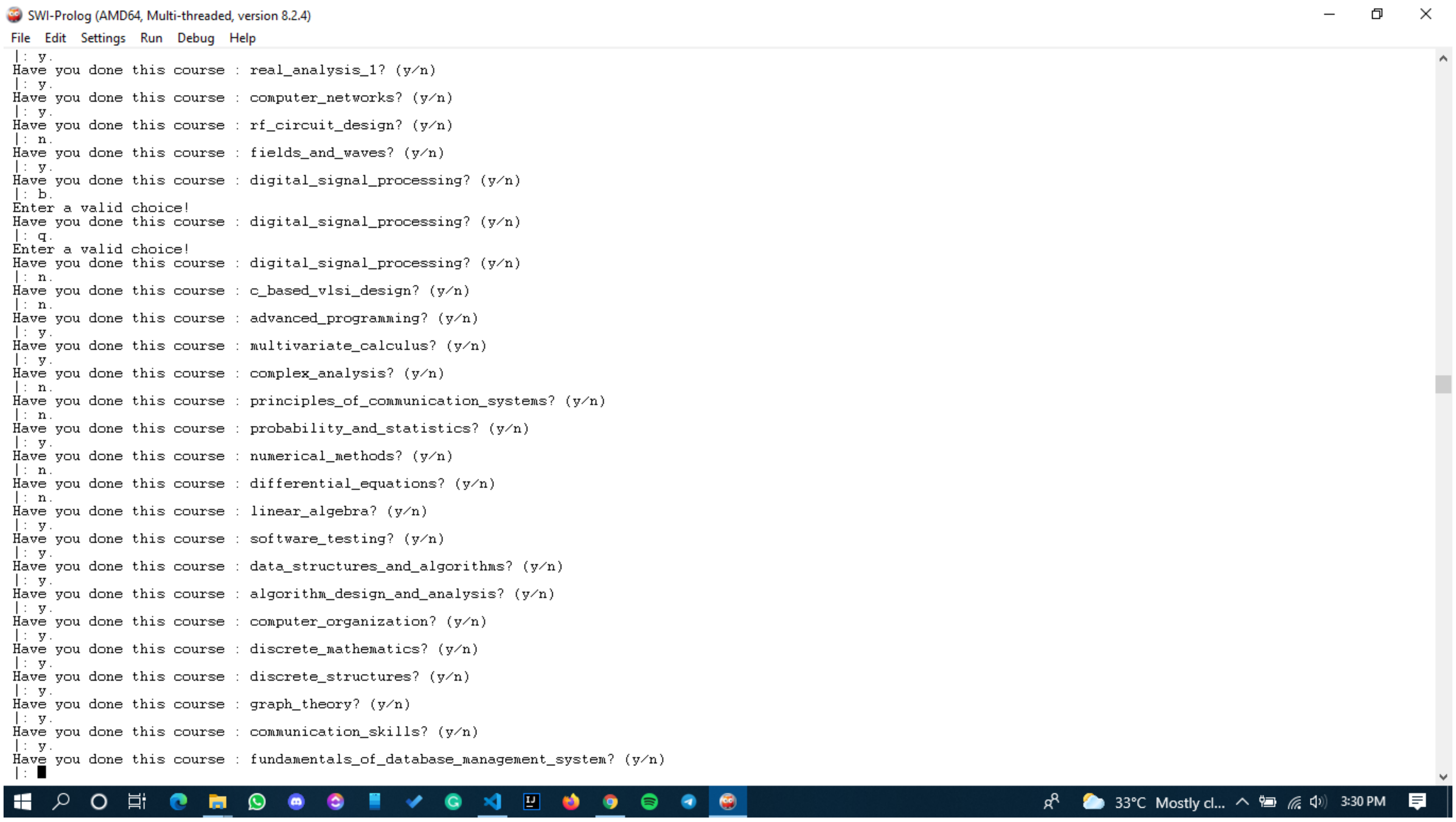
```
Sort its elements according to their priority numbers using merge
sort algorithm:
Priority number of Course = Y, where courseChecked(Course, Y)
Store the sorted list as Sorted.
If X = 1 then:
      Condition1: For Course1 and Course2 in List, the course/skill
will be considered greater than career/job. That is, push careers
towards the end of the list.
      Enforce Condition1
Else if X = 2 then:
      Do not enforce Condition1
```

Sample Input/ Output:

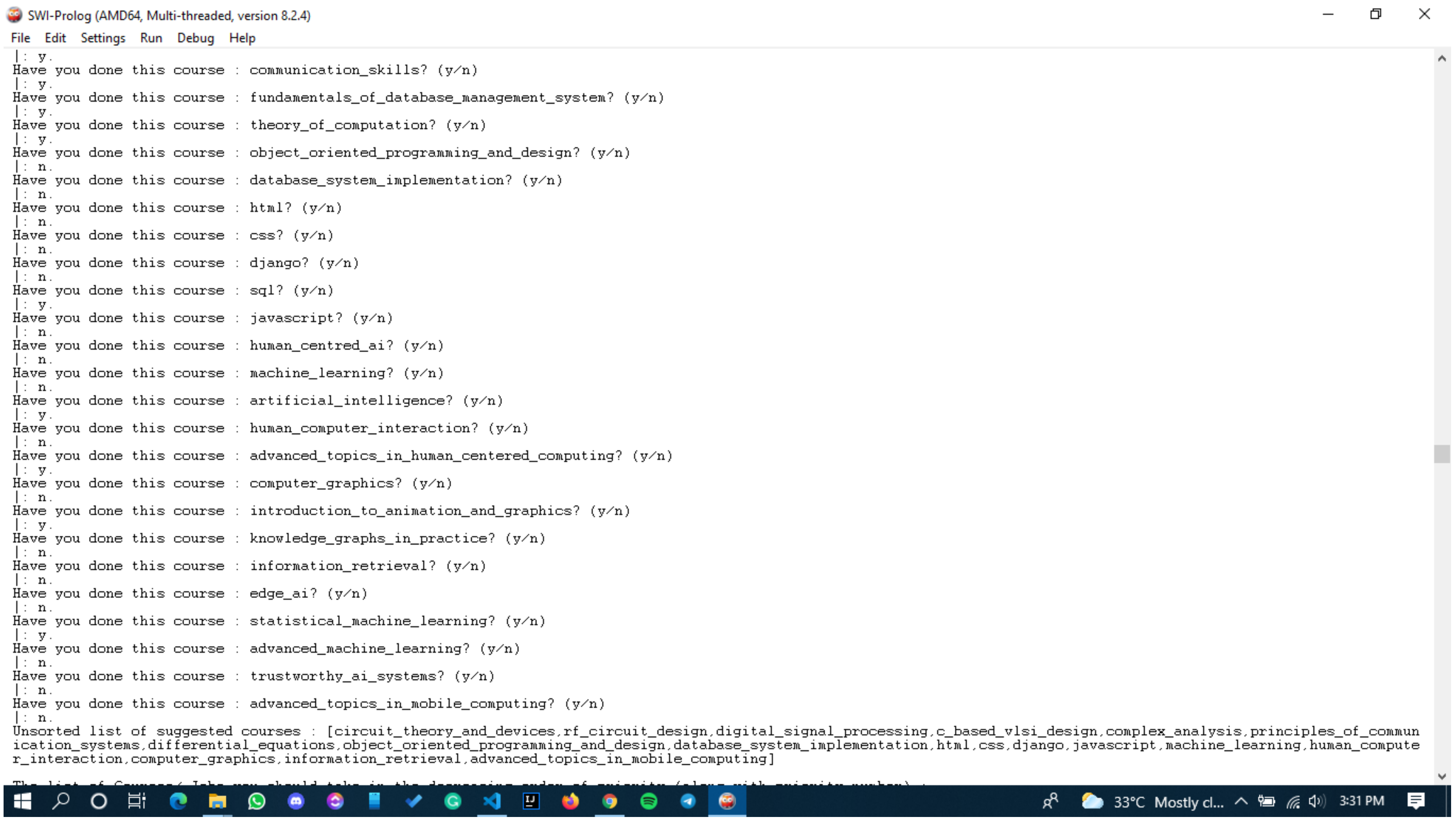
1.


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
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).

?-
%  courses.txt compiled 0.00 sec, 192 clauses
%  careers.txt compiled 0.00 sec, 9 clauses
Warning: c:/users/the kiryals/desktop/backup/ai-a1-siddharth singh kiryal-2019277/prerequisites.txt:163:
Warning:    Clauses of preRequisite/2 are not together in the source-file
Warning:    Earlier definition at c:/users/the kiryals/desktop/backup/ai-a1-siddharth singh kiryal-2019277/prerequisites.txt:1
Warning:    Current predicate: preRequisite/1
Warning:    Use :- discontiguous preRequisite/2. to suppress this message
%  preRequisites.txt compiled 0.00 sec, 375 clauses
Warning: c:/users/the kiryals/desktop/backup/ai-a1-siddharth singh kiryal-2019277/ai-a1-siddharth singh kiryal-2019277.pl:51:
Warning:    Singleton variables: [Index]
Warning: c:/users/the kiryals/desktop/backup/ai-a1-siddharth singh kiryal-2019277/ai-a1-siddharth singh kiryal-2019277.pl:84:
Warning:    Singleton variables: [X]
Warning: c:/users/the kiryals/desktop/backup/ai-a1-siddharth singh kiryal-2019277/ai-a1-siddharth singh kiryal-2019277.pl:85:
Warning:    Singleton variables: [X]
% c:/Users/The Kiryals/Desktop/backup/AI-A1-Siddharth Singh Kiryal-2019277/AI-A1-Siddharth Singh Kiryal-2019277.pl compiled 0.02 sec, 42 clauses
?- start().
Enter the list of Careers/Courses that you are interested in (only check for registered careers; to check the list of registered careers type : career(X)):
|: [hardware_engineer, software_developer, interactive_system_designer].
Have you done this course : digital_circuits? (y/n)
|: y.
Have you done this course : operating_systems? (y/n)
|: y.
Have you done this course : computer_architecture? (y/n)
|: y.
Have you done this course : introduction_to_programming? (y/n)
|: y.
Have you done this course : circuit_theory_and_devices? (y/n)
|: n.
Have you done this course : basic_electronics? (y/n)
|: y.
Have you done this course : digital_vlsi_design? (y/n)
|: y.
Have you done this course : signals_and_systems? (y/n)
|: y.
Have you done this course : embedded_logic_design? (y/n)
|: y.
Have you done this course : real_analysis_1? (y/n)
|: y.
Have you done this course : computer_networks? (y/n)
|: y.
Have you done this course : rf_circuit_design? (y/n)
|: n.
Have you done this course : fields_and_waves? (y/n)
|:
```

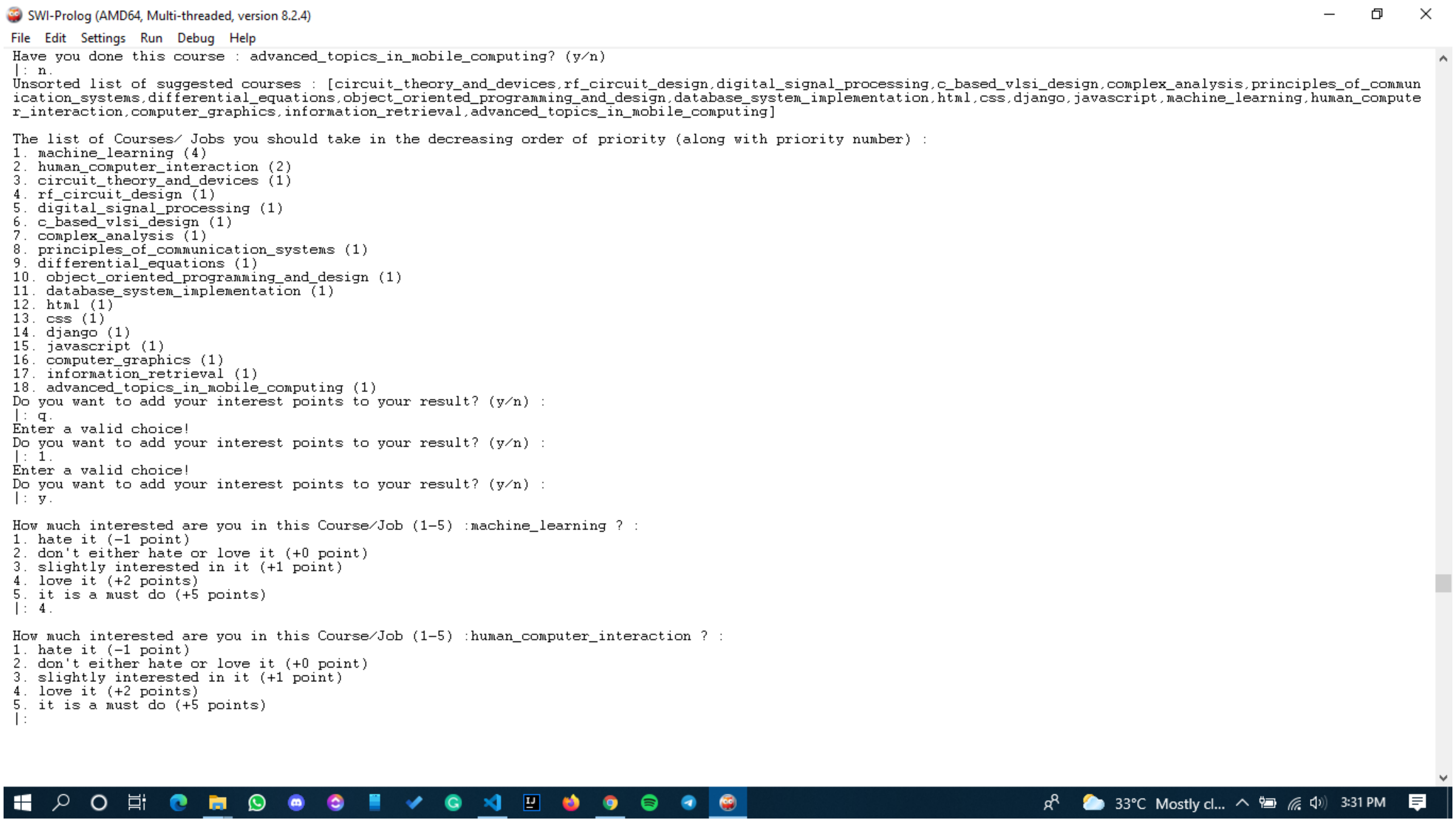

SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

```
|: y.
Have you done this course : real_analysis_1? (y/n)
|: y.
Have you done this course : computer_networks? (y/n)
|: y.
Have you done this course : rf_circuit_design? (y/n)
|: n.
Have you done this course : fields_and_waves? (y/n)
|: y.
Have you done this course : digital_signal_processing? (y/n)
|: b.
Enter a valid choice!
Have you done this course : digital_signal_processing? (y/n)
|: q.
Enter a valid choice!
Have you done this course : digital_signal_processing? (y/n)
|: n.
Have you done this course : c_based_vlsi_design? (y/n)
|: n.
Have you done this course : advanced_programming? (y/n)
|: y.
Have you done this course : multivariate_calculus? (y/n)
|: y.
Have you done this course : complex_analysis? (y/n)
|: n.
Have you done this course : principles_of_communication_systems? (y/n)
|: n.
Have you done this course : probability_and_statistics? (y/n)
|: y.
Have you done this course : numerical_methods? (y/n)
|: n.
Have you done this course : differential_equations? (y/n)
|: n.
Have you done this course : linear_algebra? (y/n)
|: y.
Have you done this course : software_testing? (y/n)
|: y.
Have you done this course : data_structures_and_algorithms? (y/n)
|: y.
Have you done this course : algorithm_design_and_analysis? (y/n)
|: y.
Have you done this course : computer_organization? (y/n)
|: y.
Have you done this course : discrete_mathematics? (y/n)
|: y.
Have you done this course : discrete_structures? (y/n)
|: y.
Have you done this course : graph_theory? (y/n)
|: y.
Have you done this course : communication_skills? (y/n)
|: y.
Have you done this course : fundamentals_of_database_management_system? (y/n)
|: 
```

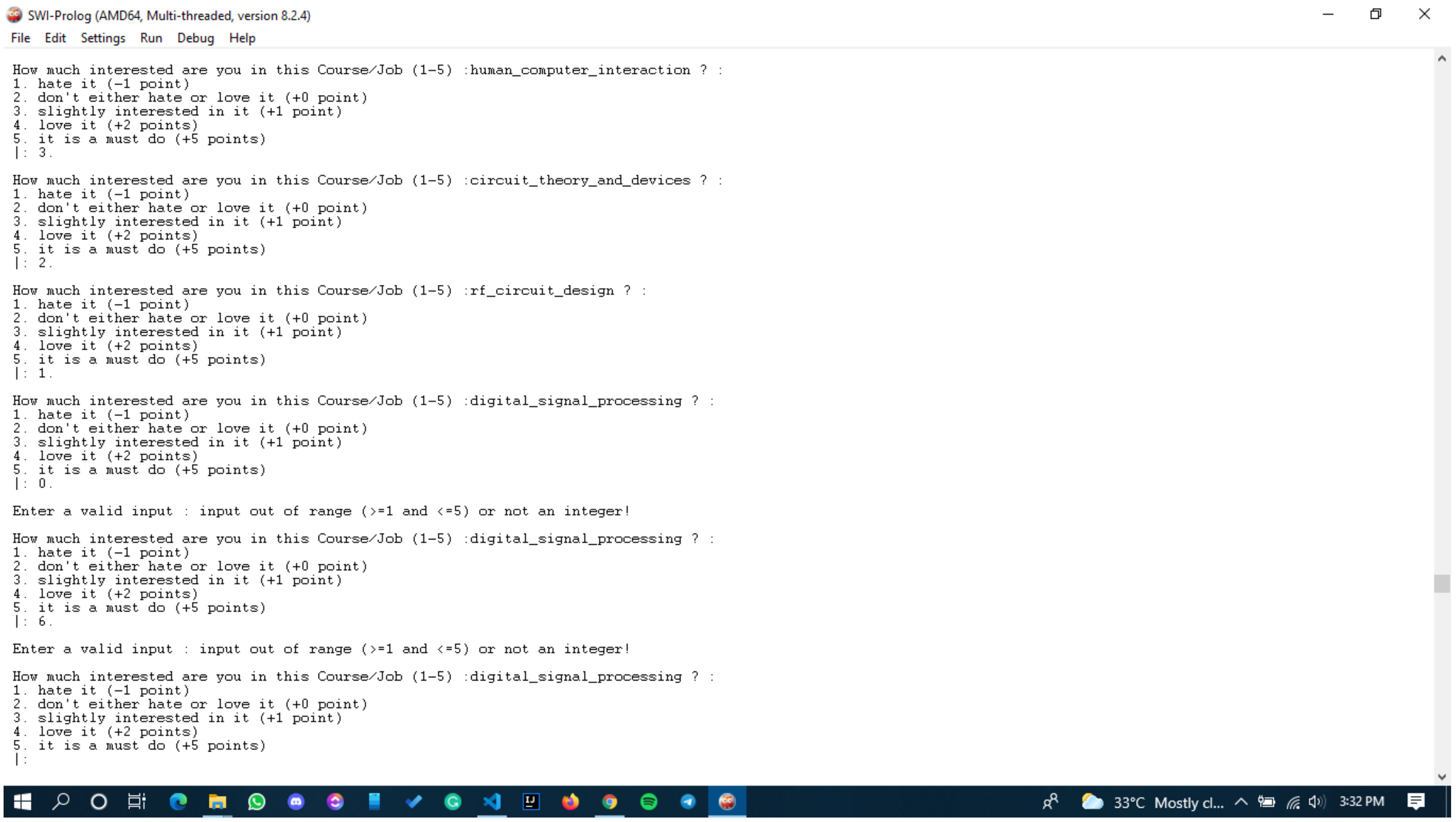


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

```
|: y.
Have you done this course : communication_skills? (y/n)
|: y.
Have you done this course : fundamentals_of_database_management_system? (y/n)
|: y.
Have you done this course : theory_of_computation? (y/n)
|: y.
Have you done this course : object_oriented_programming_and_design? (y/n)
|: n.
Have you done this course : database_system_implementation? (y/n)
|: n.
Have you done this course : html? (y/n)
|: n.
Have you done this course : css? (y/n)
|: n.
Have you done this course : django? (y/n)
|: n.
Have you done this course : sql? (y/n)
|: y.
Have you done this course : javascript? (y/n)
|: n.
Have you done this course : human_centred_ai? (y/n)
|: n.
Have you done this course : machine_learning? (y/n)
|: n.
Have you done this course : artificial_intelligence? (y/n)
|: y.
Have you done this course : human_computer_interaction? (y/n)
|: n.
Have you done this course : advanced_topics_in_human_centered_computing? (y/n)
|: y.
Have you done this course : computer_graphics? (y/n)
|: n.
Have you done this course : introduction_to_animation_and_graphics? (y/n)
|: y.
Have you done this course : knowledge_graphs_in_practice? (y/n)
|: n.
Have you done this course : information_retrieval? (y/n)
|: n.
Have you done this course : edge_ai? (y/n)
|: n.
Have you done this course : statistical_machine_learning? (y/n)
|: y.
Have you done this course : advanced_machine_learning? (y/n)
|: n.
Have you done this course : trustworthy_ai_systems? (y/n)
|: n.
Have you done this course : advanced_topics_in_mobile_computing? (y/n)
|: n.
Unsorted list of suggested courses : [circuit_theory_and_devices,rf_circuit_design,digital_signal_processing,c_based_vlsi_design,complex_analysis,principles_of_commun
ication_systems,differential_equations,object_oriented_programming_and_design,database_system_implementation,html,css,django,javascript,machine_learning,human_compute
r_interaction,computer_graphics,information_retrieval,advanced_topics_in_mobile_computing]

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
```



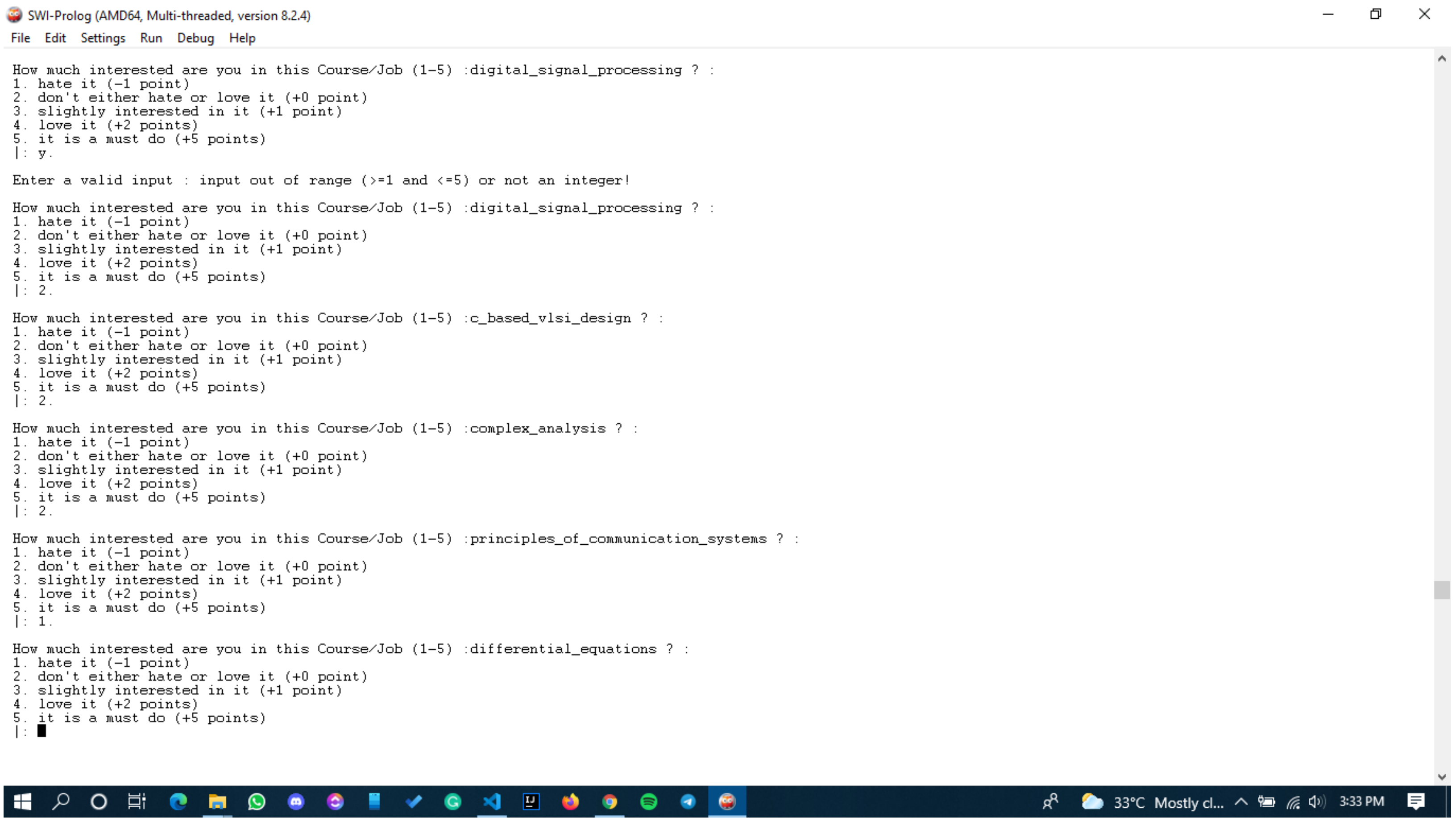
```
Have you done this course : advanced_topics_in_mobile_computing? (y/n)
|: n.
Unsorted list of suggested courses : [circuit_theory_and_devices,rf_circuit_design,digital_signal_processing,c_based_vlsi_design,complex_analysis,principles_of_commun
ication_systems,differential_equations,object_oriented_programming_and_design,database_system_implementation,html,css,django,javascript,machine_learning,human_compute
r_interaction,computer_graphics,information_retrieval,advanced_topics_in_mobile_computing]

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
1. machine_learning (4)
2. human_computer_interaction (2)
3. circuit_theory_and_devices (1)
4. rf_circuit_design (1)
5. digital_signal_processing (1)
6. c_based_vlsi_design (1)
7. complex_analysis (1)
8. principles_of_communication_systems (1)
9. differential_equations (1)
10. object_oriented_programming_and_design (1)
11. database_system_implementation (1)
12. html (1)
13. css (1)
14. django (1)
15. javascript (1)
16. computer_graphics (1)
17. information_retrieval (1)
18. advanced_topics_in_mobile_computing (1)
Do you want to add your interest points to your result? (y/n) :
|: q.
Enter a valid choice!
Do you want to add your interest points to your result? (y/n) :
|: 1.
Enter a valid choice!
Do you want to add your interest points to your result? (y/n) :
|: y.

How much interested are you in this Course/Job (1-5) :machine_learning ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 4.

How much interested are you in this Course/Job (1-5) :human_computer_interaction ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|:
```


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

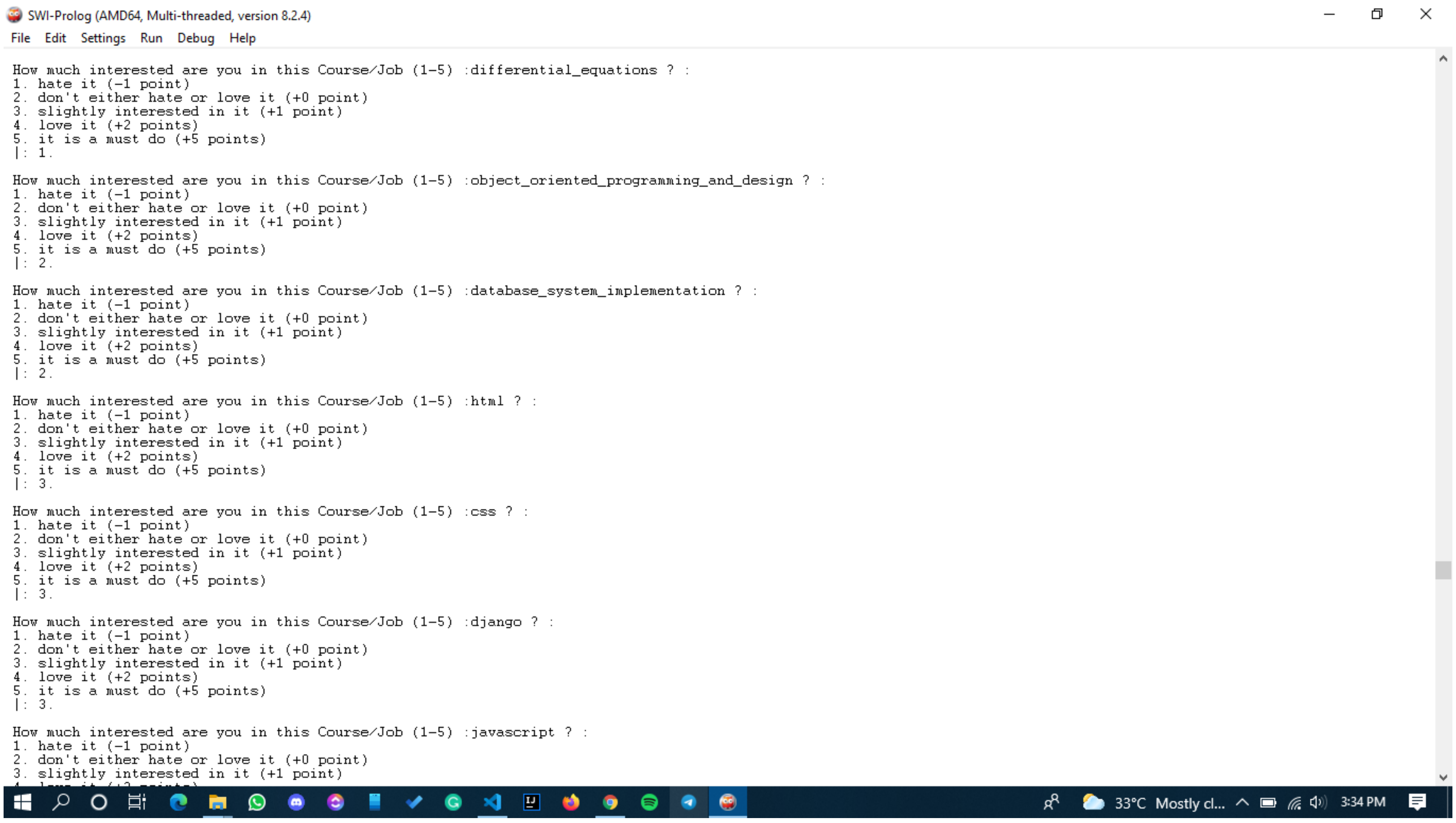
File Edit Settings Run Debug Help

```
How much interested are you in this Course/Job (1-5) :human_computer_interaction ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :circuit_theory_and_devices ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :rf_circuit_design ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 1.

How much interested are you in this Course/Job (1-5) :digital_signal_processing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 0.

Enter a valid input : input out of range (>=1 and <=5) or not an integer!

How much interested are you in this Course/Job (1-5) :digital_signal_processing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 6.

Enter a valid input : input out of range (>=1 and <=5) or not an integer!

How much interested are you in this Course/Job (1-5) :digital_signal_processing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|:
```

33°C Mostly cl... 3:32 PM

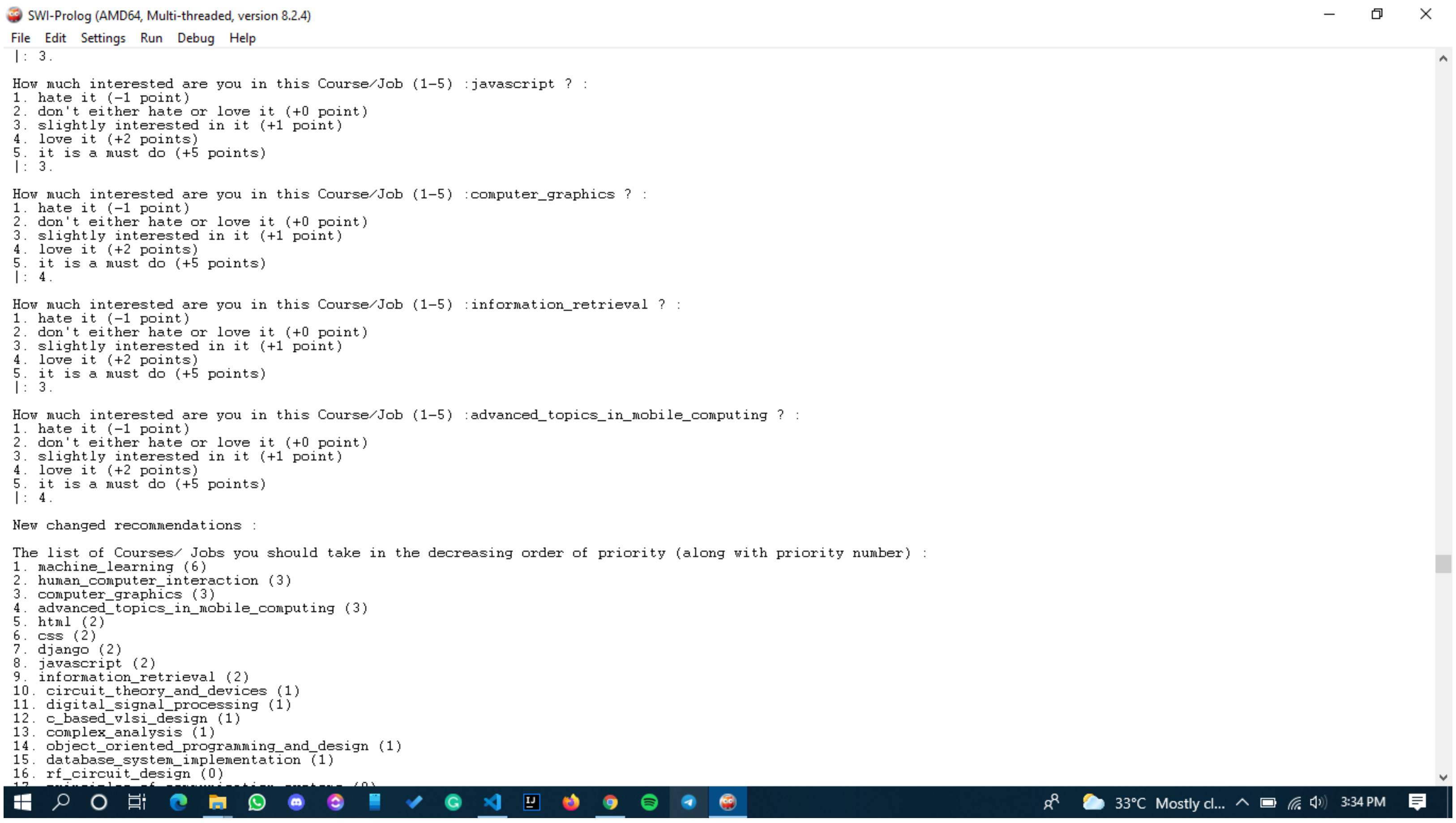

SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help

```
How much interested are you in this Course/Job (1-5) :digital_signal_processing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: y.

Enter a valid input : input out of range (>=1 and <=5) or not an integer!

How much interested are you in this Course/Job (1-5) :digital_signal_processing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :c_based_vlsi_design ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :complex_analysis ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :principles_of_communication_systems ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 1.

How much interested are you in this Course/Job (1-5) :differential_equations ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 
```

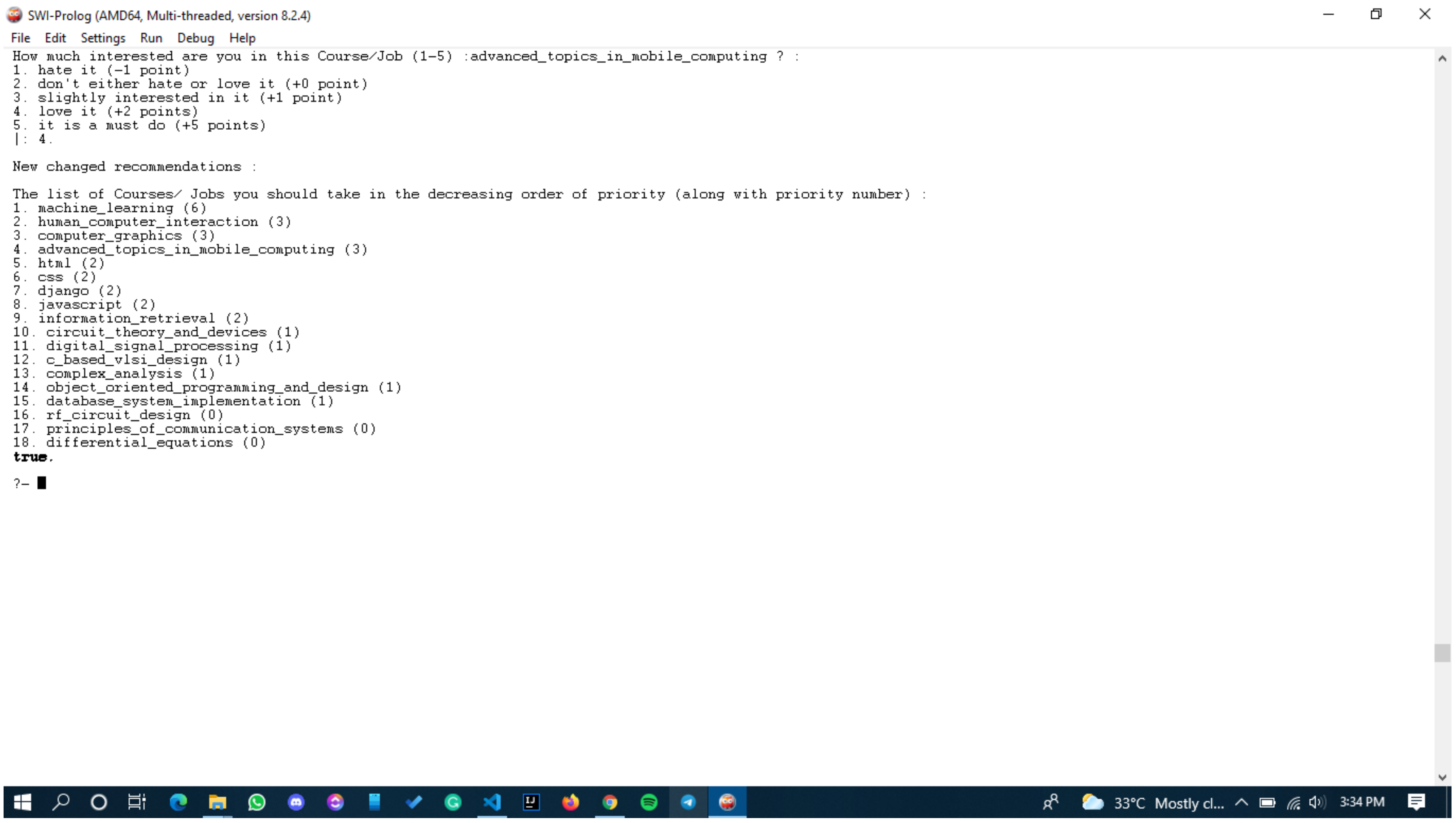

SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

File Edit Settings Run Debug Help

```
How much interested are you in this Course/Job (1-5) :differential_equations ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 1.

How much interested are you in this Course/Job (1-5) :object_oriented_programming_and_design ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :database_system_implementation ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :html ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :css ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :django ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :javascript ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
```


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help

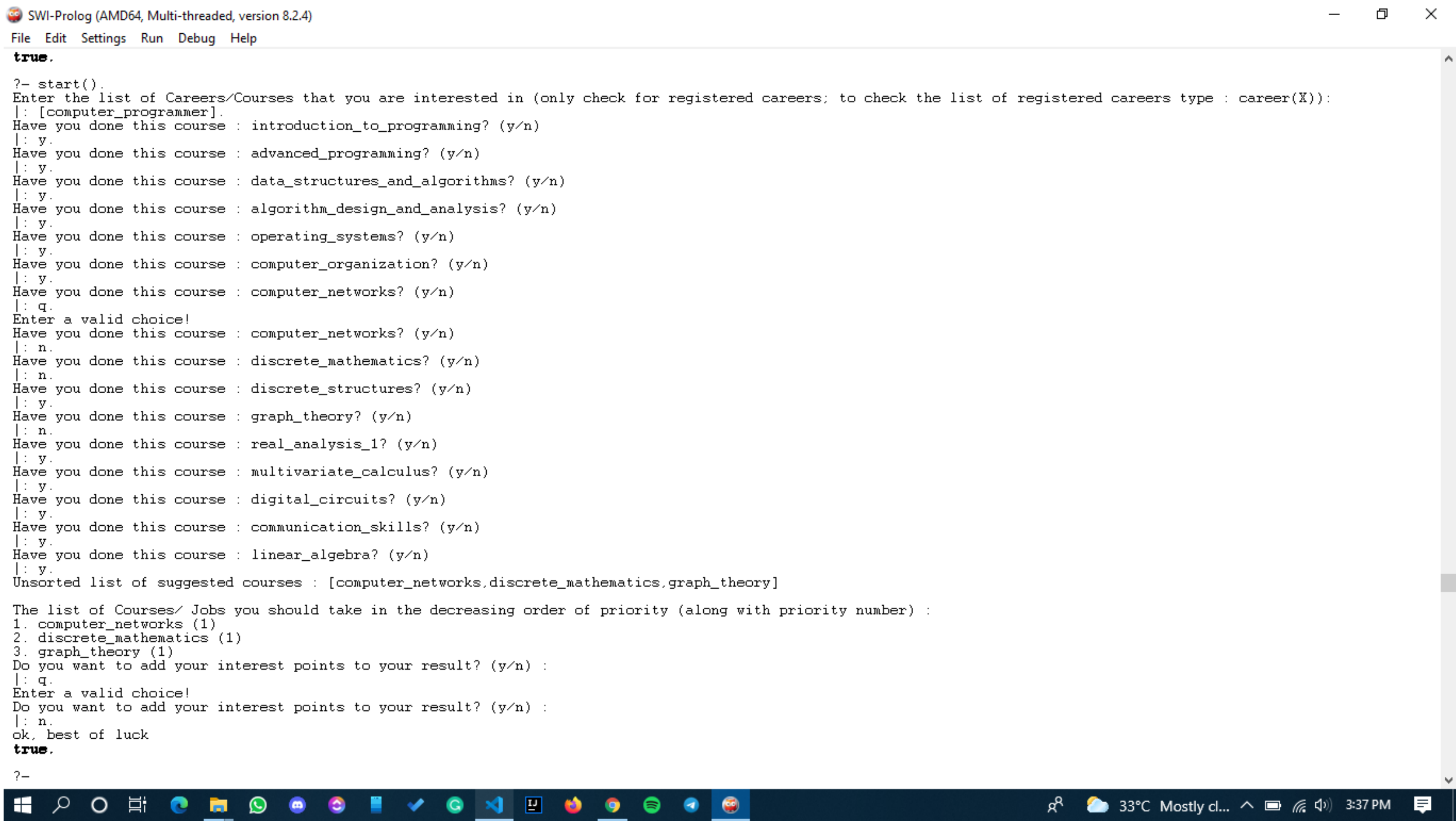
```
|: 3.

How much interested are you in this Course/Job (1-5) :javascript ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :computer_graphics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 4.

How much interested are you in this Course/Job (1-5) :information_retrieval ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :advanced_topics_in_mobile_computing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 4.

New changed recommendations :

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
1. machine_learning (6)
2. human_computer_interaction (3)
3. computer_graphics (3)
4. advanced_topics_in_mobile_computing (3)
5. html (2)
6. css (2)
7. django (2)
8. javascript (2)
9. information_retrieval (2)
10. circuit_theory_and_devices (1)
11. digital_signal_processing (1)
12. c_based_vlsi_design (1)
13. complex_analysis (1)
14. object_oriented_programming_and_design (1)
15. database_system_implementation (1)
16. rf_circuit_design (0)
```


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

File Edit Settings Run Debug Help

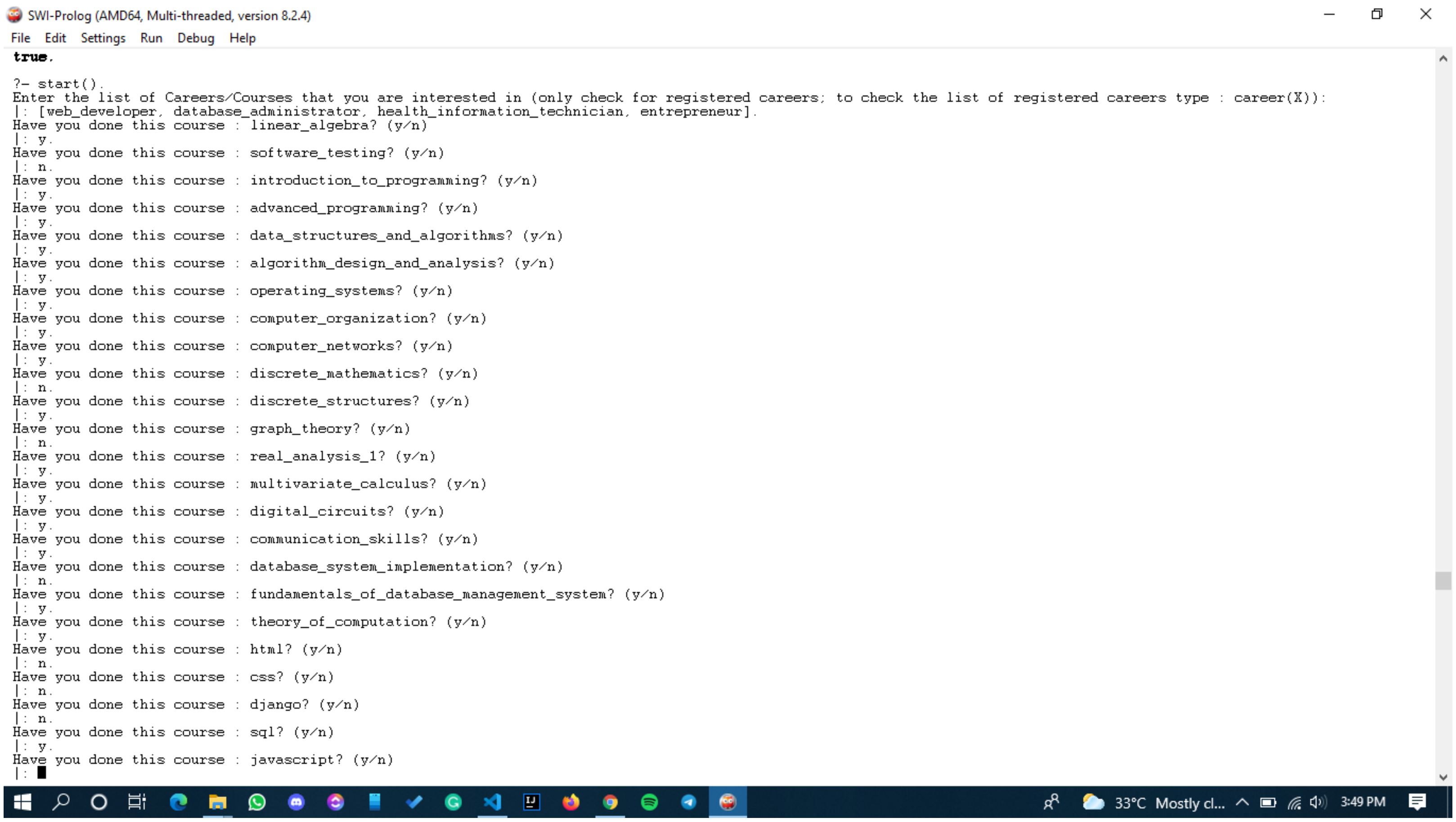
```
How much interested are you in this Course/Job (1-5) :advanced_topics_in_mobile_computing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 4.

New changed recommendations :

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
1. machine_learning (6)
2. human_computer_interaction (3)
3. computer_graphics (3)
4. advanced_topics_in_mobile_computing (3)
5. html (2)
6. css (2)
7. django (2)
8. javascript (2)
9. information_retrieval (2)
10. circuit_theory_and_devices (1)
11. digital_signal_processing (1)
12. c_based_vlsi_design (1)
13. complex_analysis (1)
14. object_oriented_programming_and_design (1)
15. database_system_implementation (1)
16. rf_circuit_design (0)
17. principles_of_communication_systems (0)
18. differential_equations (0)
true.

?-
```

2.



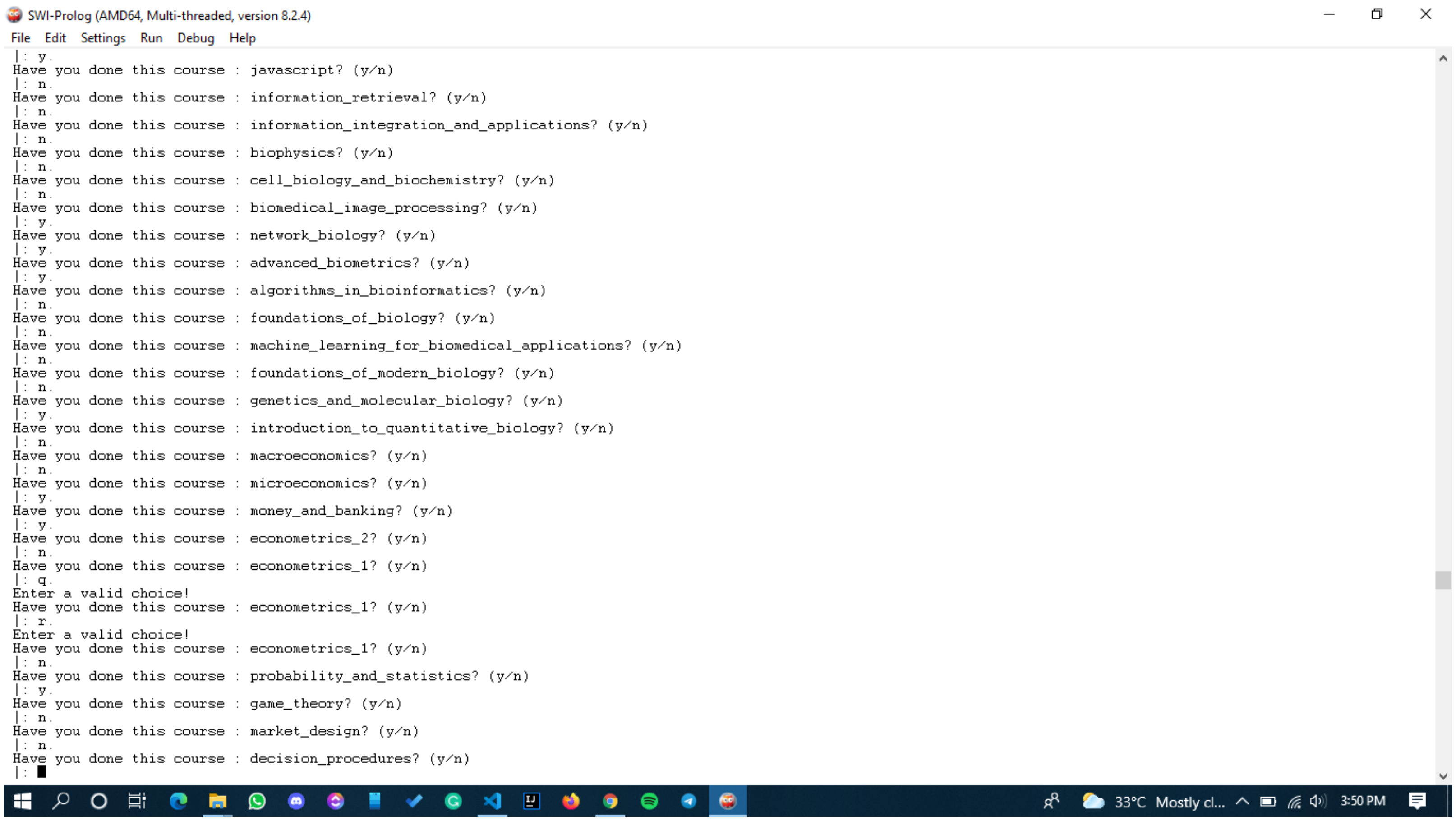
```
true.

?- start().
Enter the list of Careers/Courses that you are interested in (only check for registered careers; to check the list of registered careers type : career(X)):
|: [computer_programmer].
Have you done this course : introduction_to_programming? (y/n)
|: y.
Have you done this course : advanced_programming? (y/n)
|: y.
Have you done this course : data_structures_and_algorithms? (y/n)
|: y.
Have you done this course : algorithm_design_and_analysis? (y/n)
|: y.
Have you done this course : operating_systems? (y/n)
|: y.
Have you done this course : computer_organization? (y/n)
|: y.
Have you done this course : computer_networks? (y/n)
|: q.
Enter a valid choice!
Have you done this course : computer_networks? (y/n)
|: n.
Have you done this course : discrete_mathematics? (y/n)
|: n.
Have you done this course : discrete_structures? (y/n)
|: y.
Have you done this course : graph_theory? (y/n)
|: n.
Have you done this course : real_analysis_1? (y/n)
|: y.
Have you done this course : multivariate_calculus? (y/n)
|: y.
Have you done this course : digital_circuits? (y/n)
|: y.
Have you done this course : communication_skills? (y/n)
|: y.
Have you done this course : linear_algebra? (y/n)
|: y.
Unsorted list of suggested courses : [computer_networks,discrete_mathematics,graph_theory]

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
1. computer_networks (1)
2. discrete_mathematics (1)
3. graph_theory (1)
Do you want to add your interest points to your result? (y/n) :
|: q.
Enter a valid choice!
Do you want to add your interest points to your result? (y/n) :
|: n.
ok, best of luck
true.

?-
```

3.



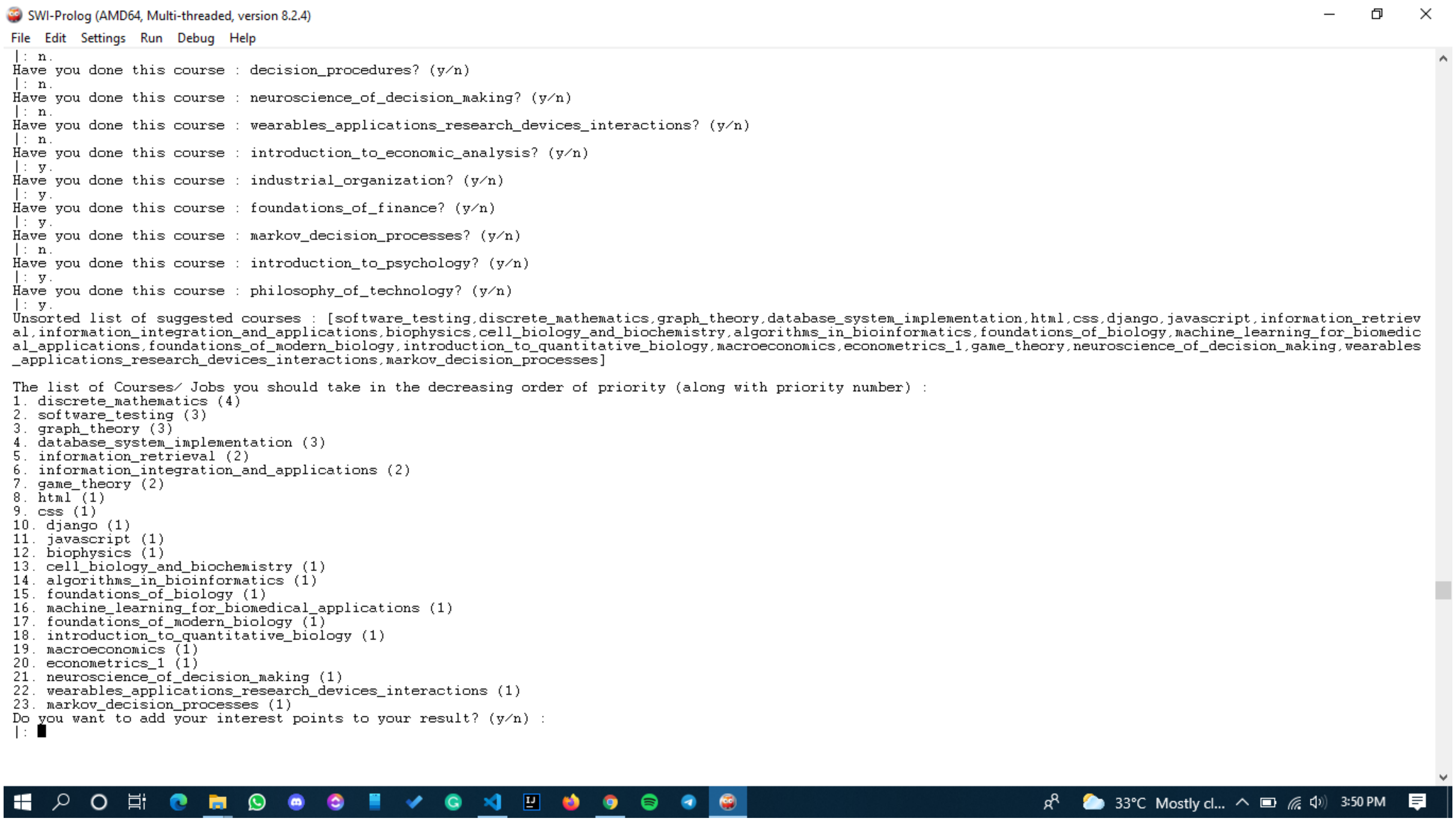
SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

```
true.

?- start().
Enter the list of Careers/Courses that you are interested in (only check for registered careers; to check the list of registered careers type : career(X)):
|: [web_developer, database_administrator, health_information_technician, entrepreneur].
Have you done this course : linear_algebra? (y/n)
|: y.
Have you done this course : software_testing? (y/n)
|: n.
Have you done this course : introduction_to_programming? (y/n)
|: y.
Have you done this course : advanced_programming? (y/n)
|: y.
Have you done this course : data_structures_and_algorithms? (y/n)
|: y.
Have you done this course : algorithm_design_and_analysis? (y/n)
|: y.
Have you done this course : operating_systems? (y/n)
|: y.
Have you done this course : computer_organization? (y/n)
|: y.
Have you done this course : computer_networks? (y/n)
|: y.
Have you done this course : discrete_mathematics? (y/n)
|: n.
Have you done this course : discrete_structures? (y/n)
|: y.
Have you done this course : graph_theory? (y/n)
|: n.
Have you done this course : real_analysis_1? (y/n)
|: y.
Have you done this course : multivariate_calculus? (y/n)
|: y.
Have you done this course : digital_circuits? (y/n)
|: y.
Have you done this course : communication_skills? (y/n)
|: y.
Have you done this course : database_system_implementation? (y/n)
|: n.
Have you done this course : fundamentals_of_database_management_system? (y/n)
|: y.
Have you done this course : theory_of_computation? (y/n)
|: y.
Have you done this course : html? (y/n)
|: n.
Have you done this course : css? (y/n)
|: n.
Have you done this course : django? (y/n)
|: n.
Have you done this course : sql? (y/n)
|: y.
Have you done this course : javascript? (y/n)
|: 
```



SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

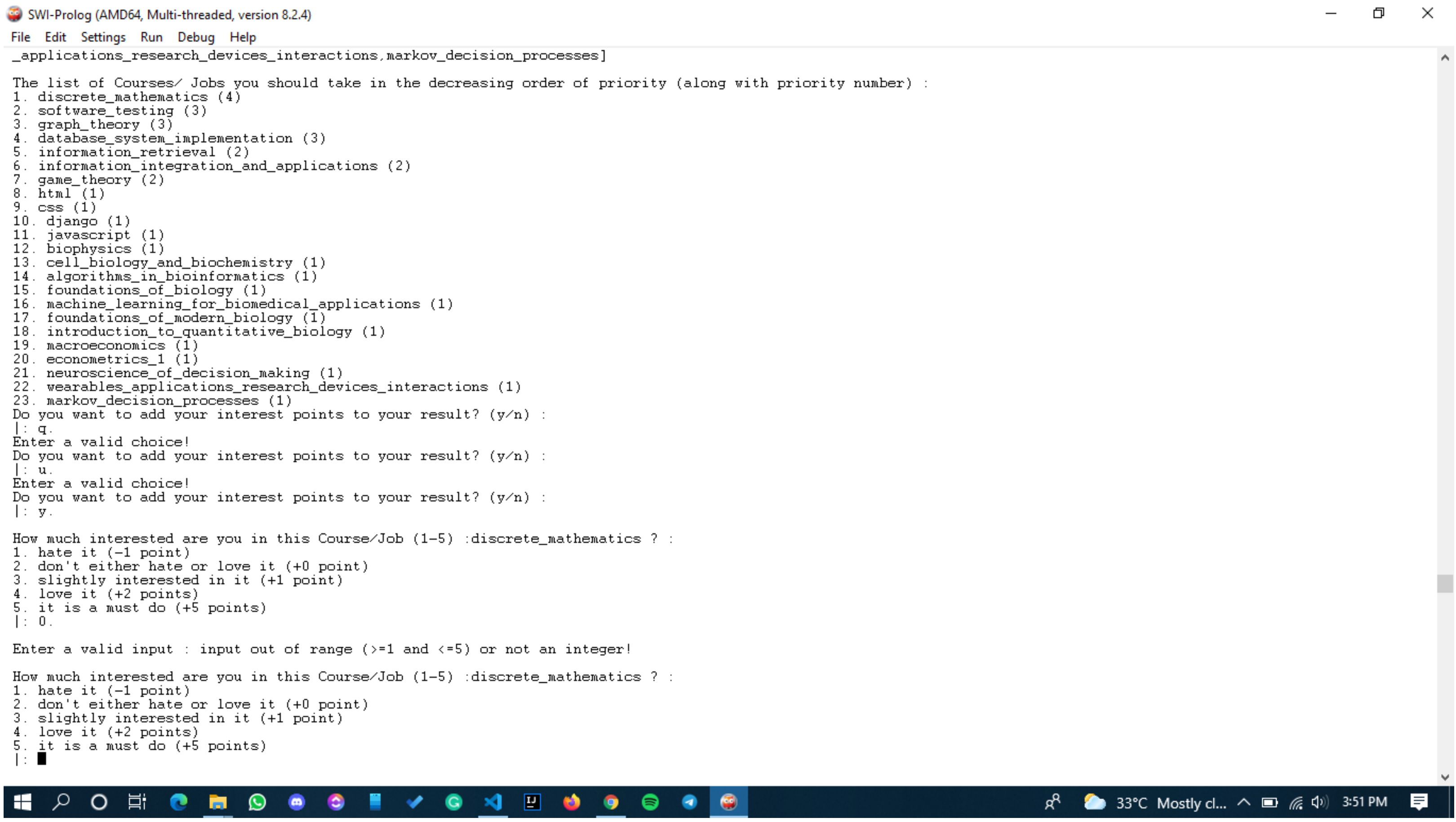
```
|: y.
Have you done this course : javascript? (y/n)
|: n.
Have you done this course : information_retrieval? (y/n)
|: n.
Have you done this course : information_integration_and_applications? (y/n)
|: n.
Have you done this course : biophysics? (y/n)
|: n.
Have you done this course : cell_biology_and_biochemistry? (y/n)
|: n.
Have you done this course : biomedical_image_processing? (y/n)
|: y.
Have you done this course : network_biology? (y/n)
|: y.
Have you done this course : advanced_biometrics? (y/n)
|: y.
Have you done this course : algorithms_in_bioinformatics? (y/n)
|: n.
Have you done this course : foundations_of_biology? (y/n)
|: n.
Have you done this course : machine_learning_for_biomedical_applications? (y/n)
|: n.
Have you done this course : foundations_of_modern_biology? (y/n)
|: n.
Have you done this course : genetics_and_molecular_biology? (y/n)
|: y.
Have you done this course : introduction_to_quantitative_biology? (y/n)
|: n.
Have you done this course : macroeconomics? (y/n)
|: n.
Have you done this course : microeconomics? (y/n)
|: y.
Have you done this course : money_and_banking? (y/n)
|: y.
Have you done this course : econometrics_2? (y/n)
|: n.
Have you done this course : econometrics_1? (y/n)
|: q.
Enter a valid choice!
Have you done this course : econometrics_1? (y/n)
|: r.
Enter a valid choice!
Have you done this course : econometrics_1? (y/n)
|: n.
Have you done this course : probability_and_statistics? (y/n)
|: y.
Have you done this course : game_theory? (y/n)
|: n.
Have you done this course : market_design? (y/n)
|: n.
Have you done this course : decision_procedures? (y/n)
|:
```



SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

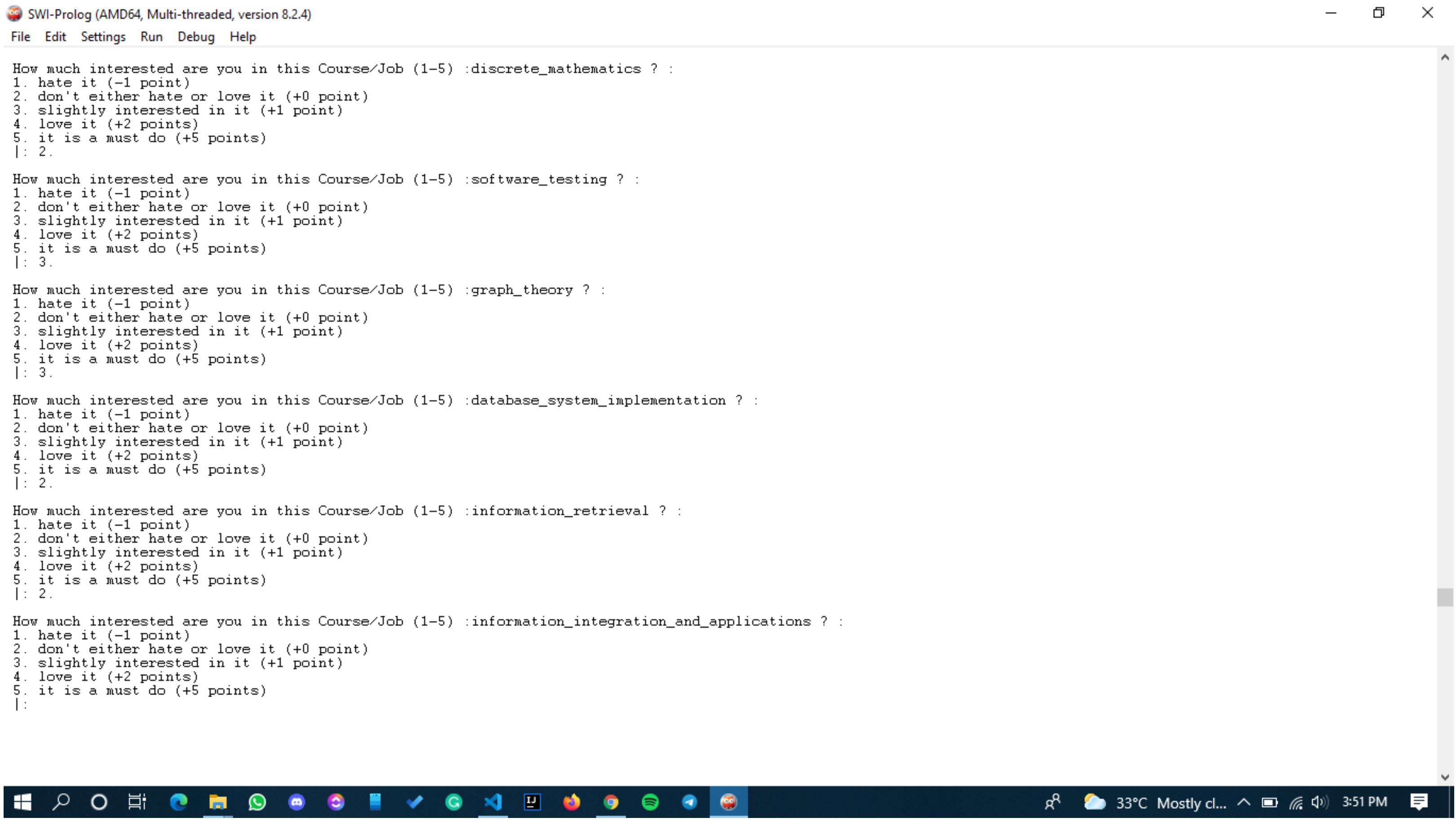
File Edit Settings Run Debug Help

```
|: n.
Have you done this course : decision_procedures? (y/n)
|: n.
Have you done this course : neuroscience_of_decision_making? (y/n)
|: n.
Have you done this course : wearables_applications_research_devices_interactions? (y/n)
|: n.
Have you done this course : introduction_to_economic_analysis? (y/n)
|: y.
Have you done this course : industrial_organization? (y/n)
|: y.
Have you done this course : foundations_of_finance? (y/n)
|: y.
Have you done this course : markov_decision_processes? (y/n)
|: n.
Have you done this course : introduction_to_psychology? (y/n)
|: y.
Have you done this course : philosophy_of_technology? (y/n)
|: y.
Unsorted list of suggested courses : [software_testing,discrete_mathematics,graph_theory,database_system_implementation,html,css,django,javascript,information_retriev
al,information_integration_and_applications,biophysics,cell_biology_and_biochemistry,algorithms_in_bioinformatics,foundations_of_biology,machine_learning_for_biomedic
al_applications,foundations_of_modern_biology,introduction_to_quantitative_biology,macroeconomics,econometrics_1,game_theory,neuroscience_of_decision_making,wearables
_applications_research_devices_interactions,markov_decision_processes]

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
1. discrete_mathematics (4)
2. software_testing (3)
3. graph_theory (3)
4. database_system_implementation (3)
5. information_retrieval (2)
6. information_integration_and_applications (2)
7. game_theory (2)
8. html (1)
9. css (1)
10. django (1)
11. javascript (1)
12. biophysics (1)
13. cell_biology_and_biochemistry (1)
14. algorithms_in_bioinformatics (1)
15. foundations_of_biology (1)
16. machine_learning_for_biomedical_applications (1)
17. foundations_of_modern_biology (1)
18. introduction_to_quantitative_biology (1)
19. macroeconomics (1)
20. econometrics_1 (1)
21. neuroscience_of_decision_making (1)
22. wearables_applications_research_devices_interactions (1)
23. markov_decision_processes (1)
Do you want to add your interest points to your result? (y/n) :
|: 
```



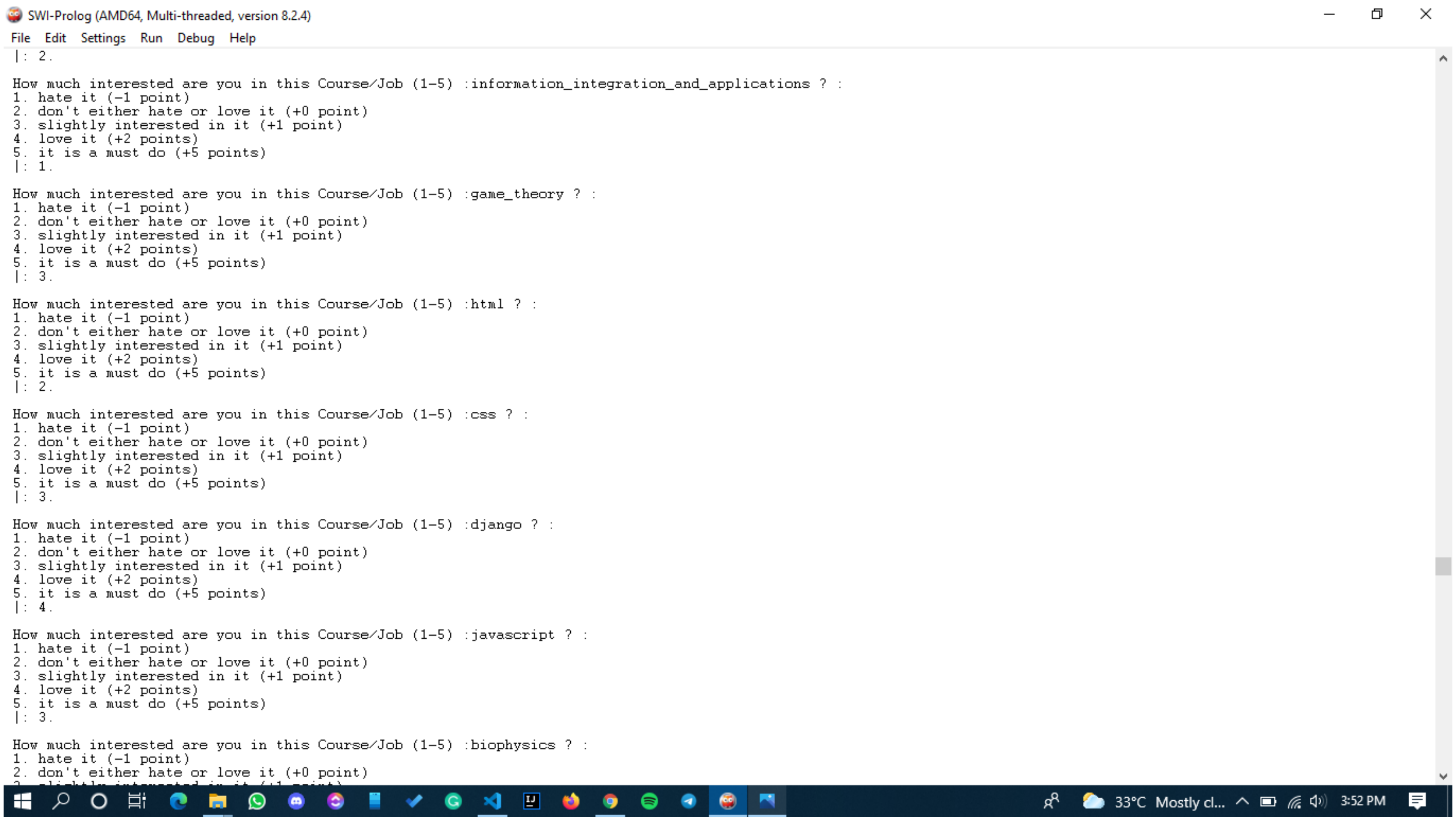
SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

```
_applications_research_devices_interactions,markov_decision_processes]

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
1. discrete_mathematics (4)
2. software_testing (3)
3. graph_theory (3)
4. database_system_implementation (3)
5. information_retrieval (2)
6. information_integration_and_applications (2)
7. game_theory (2)
8. html (1)
9. css (1)
10. django (1)
11. javascript (1)
12. biophysics (1)
13. cell_biology_and_biochemistry (1)
14. algorithms_in_bioinformatics (1)
15. foundations_of_biology (1)
16. machine_learning_for_biomedical_applications (1)
17. foundations_of_modern_biology (1)
18. introduction_to_quantitative_biology (1)
19. macroeconomics (1)
20. econometrics_1 (1)
21. neuroscience_of_decision_making (1)
22. wearables_applications_research_devices_interactions (1)
23. markov_decision_processes (1)
Do you want to add your interest points to your result? (y/n) :
|: q.
Enter a valid choice!
Do you want to add your interest points to your result? (y/n) :
|: u.
Enter a valid choice!
Do you want to add your interest points to your result? (y/n) :
|: y.

How much interested are you in this Course/Job (1-5) :discrete_mathematics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 0.

Enter a valid input : input out of range (>=1 and <=5) or not an integer!

How much interested are you in this Course/Job (1-5) :discrete_mathematics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 
```

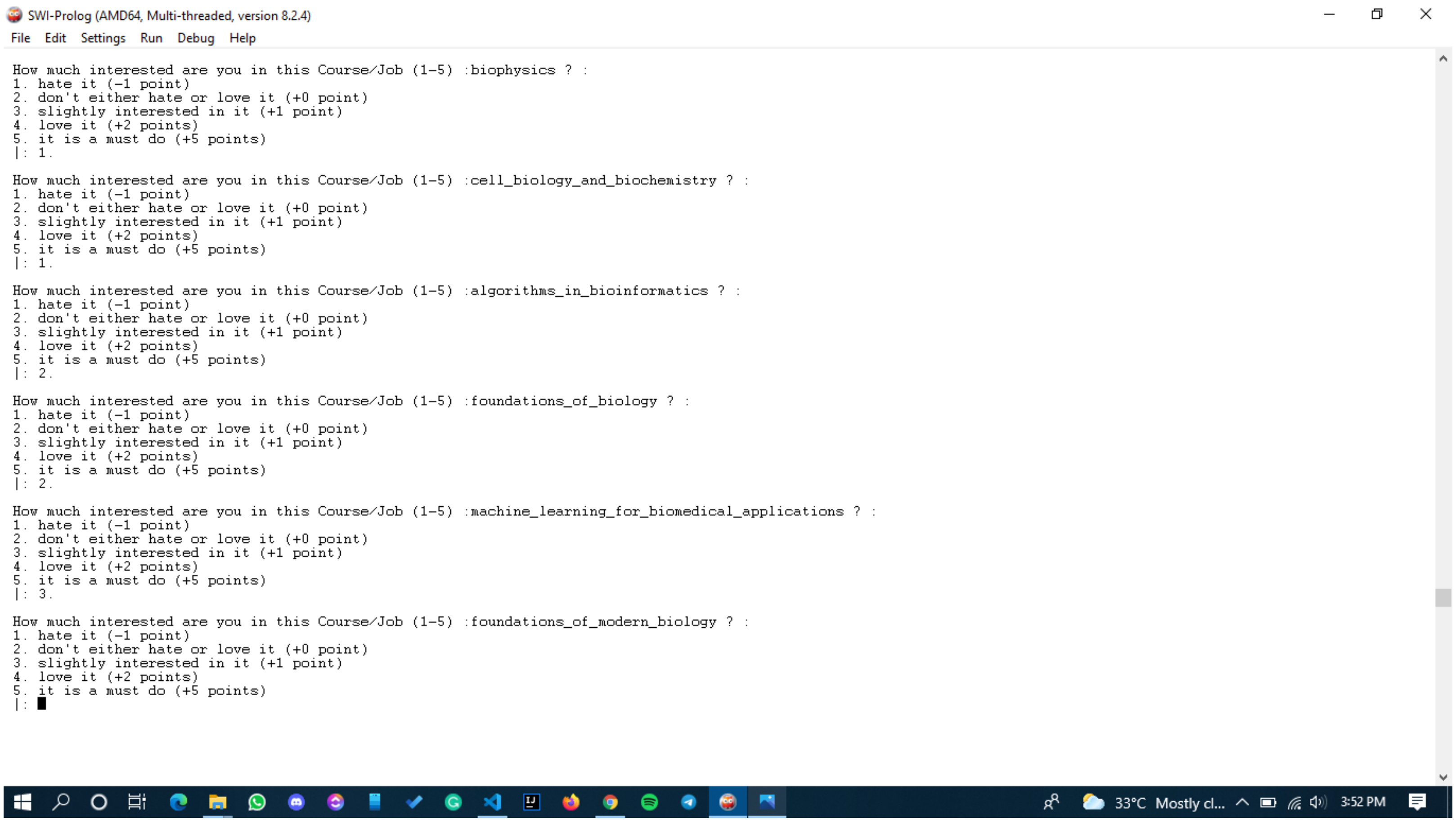


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

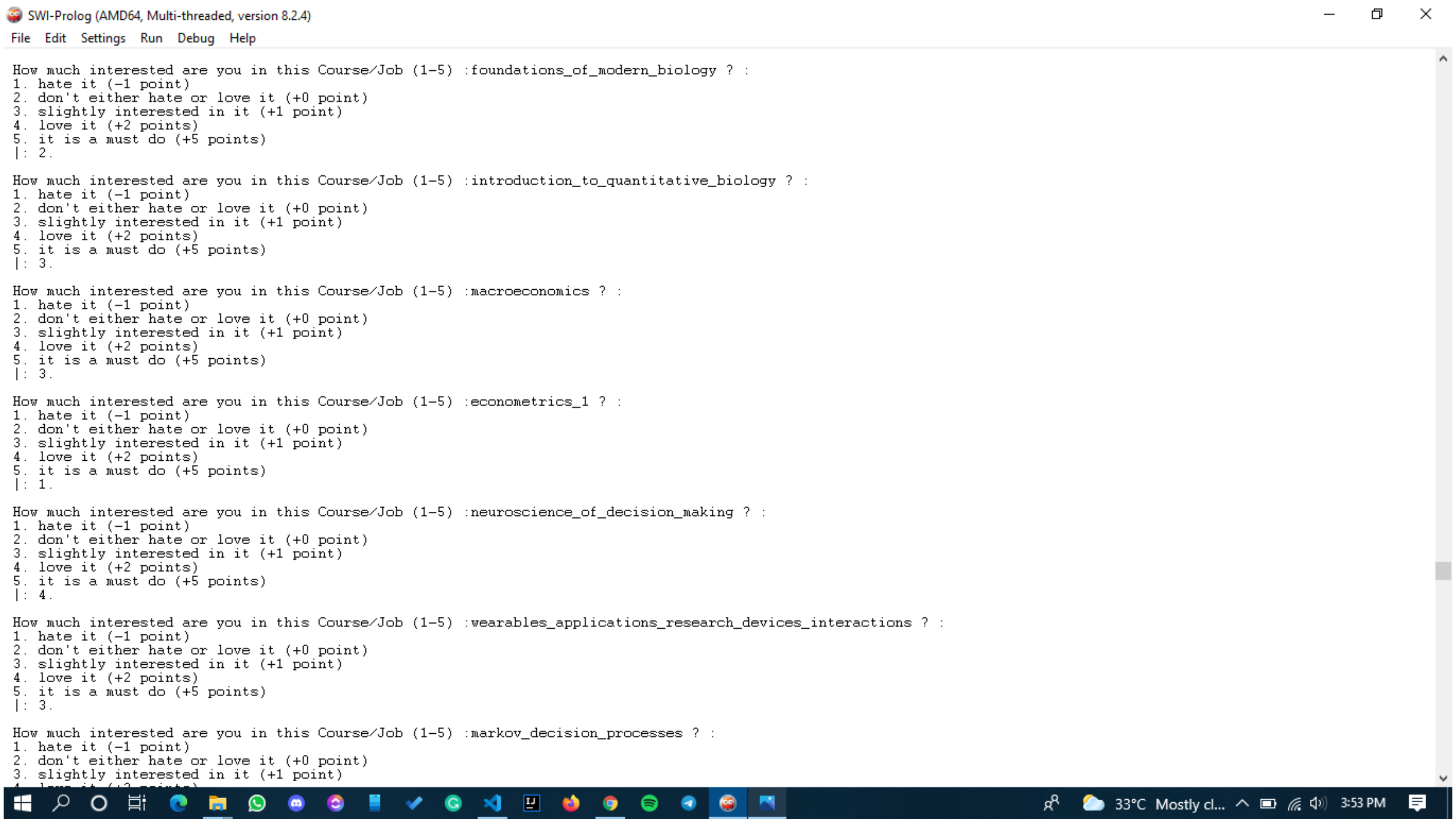
```
How much interested are you in this Course/Job (1-5) :discrete_mathematics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :software_testing ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :graph_theory ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :database_system_implementation ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :information_retrieval ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :information_integration_and_applications ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|:
```


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help

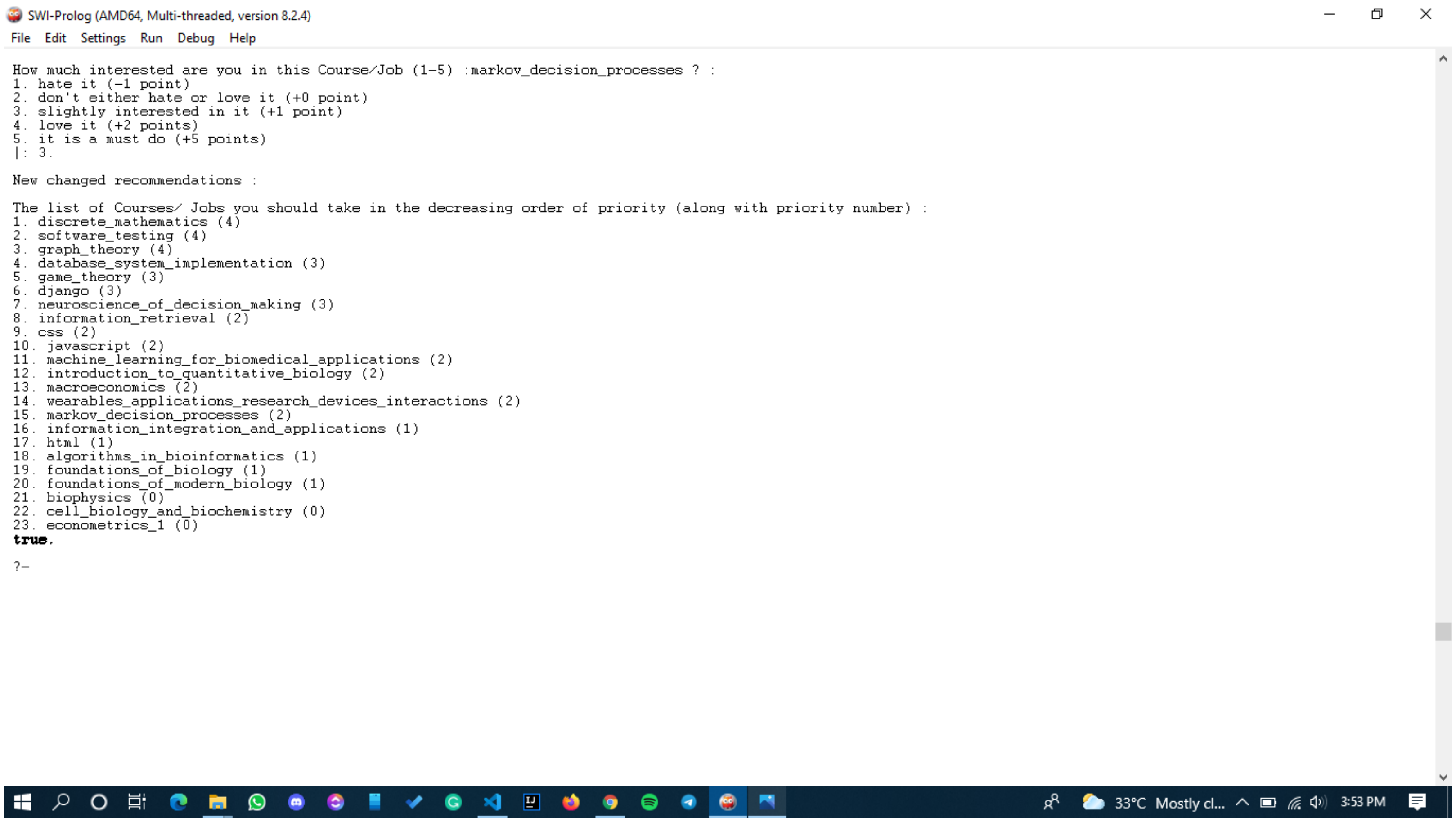
```
|: 2.

How much interested are you in this Course/Job (1-5) :information_integration_and_applications ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 1.

How much interested are you in this Course/Job (1-5) :game_theory ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :html ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :css ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :django ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 4.

How much interested are you in this Course/Job (1-5) :javascript ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :biophysics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
```


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)

```
How much interested are you in this Course/Job (1-5) :biophysics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 1.

How much interested are you in this Course/Job (1-5) :cell_biology_and_biochemistry ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 1.

How much interested are you in this Course/Job (1-5) :algorithms_in_bioinformatics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :foundations_of_biology ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :machine_learning_for_biomedical_applications ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :foundations_of_modern_biology ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 
```



```
How much interested are you in this Course/Job (1-5) :foundations_of_modern_biology ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 2.

How much interested are you in this Course/Job (1-5) :introduction_to_quantitative_biology ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :macroeconomics ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :econometrics_1 ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 1.

How much interested are you in this Course/Job (1-5) :neuroscience_of_decision_making ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 4.

How much interested are you in this Course/Job (1-5) :wearables_applications_research_devices_interactions ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

How much interested are you in this Course/Job (1-5) :markov_decision_processes ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
```


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
33°C Mostly cl...
3:53 PM

```
How much interested are you in this Course/Job (1-5) :markov_decision_processes ? :
1. hate it (-1 point)
2. don't either hate or love it (+0 point)
3. slightly interested in it (+1 point)
4. love it (+2 points)
5. it is a must do (+5 points)
|: 3.

New changed recommendations :

The list of Courses/ Jobs you should take in the decreasing order of priority (along with priority number) :
1. discrete_mathematics (4)
2. software_testing (4)
3. graph_theory (4)
4. database_system_implementation (3)
5. game_theory (3)
6. django (3)
7. neuroscience_of_decision_making (3)
8. information_retrieval (2)
9. css (2)
10. javascript (2)
11. machine_learning_for_biomedical_applications (2)
12. introduction_to_quantitative_biology (2)
13. macroeconomics (2)
14. wearables_applications_research_devices_interactions (2)
15. markov_decision_processes (2)
16. information_integration_and_applications (1)
17. html (1)
18. algorithms_in_bioinformatics (1)
19. foundations_of_biology (1)
20. foundations_of_modern_biology (1)
21. biophysics (0)
22. cell_biology_and_biochemistry (0)
23. econometrics_1 (0)
true.

?-
```
