Notes:

- This is a **take home** exam. You should treat it as you would an invigilated exam, and answer the questions using your own knowledge and without assistance from anyone. However, you may refer to the course notes and you may run any programs you are asked to write.

- Use a text editor or standard word processor (such as MSWord or Open Office) to write your answers.

- You have a maximum of five hours to answer the exam. Allow sufficient time to submit your answers; do not leave this to the last minute!

- Upload your answers to the “exam” file area on the Student Portal, [https://student.portalen.uu.se](https://student.portalen.uu.se). A plagiarism check will be made on all uploaded files.

- This exam has 12 questions. The questions are *not* of equal difficulty.

- Answer all questions. There are no optional questions.
Section A: Python

1. Describe how you can write a Python program using Swedish words in place of the English names “pickAFile”, “makePicture”, etc.
   Some words in Python can only be written in English. Give two examples.

2. Give values for \( r \), \( g \) and \( b \) in \texttt{makeColor(r, g, b)} to make a shade of pink.

3. The image below on the left was taken at Lunsentorpet, on the Uplands Trail. I would like this (along with other photographs taken in the area) to be processed so that they appear like they might in winter. For example, the image below on the right shows a possible transformation. This transformation was done by changing the green pixels into white.

   (a) Define a function \texttt{mostlyGreen} to test if a pixel has more green than either red or blue components.

   (b) Using your function above, write a function \texttt{winterize} that changes all the mostly green pixels in a picture into snow.

   For partial marks in this part, you can make all the snow pure white. However, pure white snow hides detail in the original image. For more realistic snow, you need to allow shades of grey. The shade of grey depends on the luminance of the pixel. You can calculate luminance using the formula \( 0.3r + 0.59g + 0.11b \), where \( r \), \( g \) and \( b \) are the red, green and blue levels respectively.

   Used directly, this formula will make the snow too dark, so lighten it by scaling the luminance by, say, a factor of 1.6.

You can test your solution on the images in the \texttt{/it/kurs/prognovis/images} directory. The images are also available on the course web site.

\begin{center}
\begin{tabular}{l}
\hline
\textbf{Solution:} \\
\hline
\begin{verbatim}
def mostlyGreen(p):
    return getGreen(p) > max(getRed(p), getBlue(p))+10

def winterize(pic):
    for p in getPixels(pic):
        if mostlyGreen(p):
            snow = getGreen(p) + 120
            setBlue(p, snow)
            setRed(p, snow)
            setGreen(p, snow)
\end{verbatim}
\end{tabular}
\end{center}
Section B: Prolog

4. Write a Prolog predicate \texttt{twice(X,Y)} that will succeed exactly twice, each time binding \texttt{Y} to the value \texttt{X}. For example, the predicate should behave as follows:

\begin{verbatim}
?- twice(hi, Y).
Y = hi ;
Y = hi ;
no
\end{verbatim}

\textbf{Solution:}

\begin{verbatim}
twice(X, X).
twice(X, X).
\end{verbatim}

5. Some rules for selecting a wine to accompany a meal are:

- white wine goes with fish;
- red wine goes with meat; and
- drink champagne if the meal is a celebration; however,
- only drink orange juice if you are driving.

Express these rules in Prolog by writing a predicate \texttt{wine(W)} that will select a style of drink \texttt{W} depending on which of the four facts \texttt{fish}, \texttt{meat}, \texttt{celebration} and \texttt{driving} are true.

For example, if \texttt{fact(meat)} and \texttt{fact(fish)} are both true, then \texttt{wine(W)} should return both \texttt{white} and \texttt{red}, but not \texttt{champagne} or \texttt{orange}.

\textbf{Solution:}

\begin{verbatim}
wine( orange ) :- fact( driving ).
wine( white ) :- \(+ fact( driving ), fact( fish ).
wine( red ) :- \(+ fact( driving ), fact( meat ).
wine( champagne ) :- \(+ fact(driving), fact( celebration ).
\end{verbatim}

6. A “turtle” is an electronic device that can be instructed to move around on a sheet of paper, dragging a pen. The pen draws on the paper when it is down, and does not draw when it is up.

Use Prolog grammar rules to write a parser for a language of “turtle” commands, where a command can be: “forward \texttt{(distance)}”, “turn \texttt{(direction)}”, “pen \texttt{(movement)}”, or “goto \texttt{(position)}”. \texttt{(distance)} is either nothing (indicating a default distance), or a number. \texttt{(direction)} is “left”, “right” or an angle (a number). \texttt{(movement)} is either “up” or “down”. \texttt{(position)} is an x-coordinate (a number) followed by a y-coordinate (a number). Any command can be followed by “and then \texttt{(command)}”.

For example, your parser should be able to recognise commands such as these:

- \texttt{phrase(turtle, [forward, 20]).}
- \texttt{phrase(turtle, [turn, left, and then, forward]).}

CONTINUED
• phrase(turtle. [turn, left, and, then, forward, 400, and, then, goto, 0, 0]).

Solution:

```
cmdS --> cmd, moreCmd.
moreCmd --> [and, then], cmdS.
moreCmd --> [].
cmd --> [forward], distance.
cmd --> [turn], dir.
cmd --> [pen], motion.
cmd --> [goto, X, Y], {integer(X), integer(Y)}.
distance --> [].
distance --> [N], {integer(N)}.
dir --> [left].
dir --> [right].
dir --> [N], {integer(N)}.
motion --> [up].
motion --> [down].
```

Section C: Erlang

7. Write an Erlang pattern that will result in the variable X having the value ok when matched against the tuple \{first,second,\{third,ok\}\}, but will fail to match against any other tuple.

Solution: \{first,second,\{third, X\}\}.

8. Write an Erlang term that expresses the same information as the Prolog term \(np(det(the),cat)\)

Solution: \{np,\{det.the\},cat\}

9. A recent traffic jam in China has highlighted the need to monitor the number of cars entering and leaving a roading system. Write an Erlang program that can be used to assist in traffic management.

The program should define a function with two arguments: the number of cars (load) currently in the network, and the maximum network capacity. The function should wait for “entry request” and “exiting” messages. An “entry request” should result in either a granted or denied message being sent back to the sender. If entry is granted, then the network load is increased by one. Otherwise, the number remains unchanged. An “exiting” message is sent when a car leaves the network. In this case, the network load is decreased by one (no reply message is necessary).
Section D: General

10. In three or four sentences, explain the most important differences between variables in Python, Prolog and Erlang. Do not focus on syntactic differences (i.e. the appearance of the variables), but rather on the different ways variables behave and can be used.

Solution: Python variables can be changed. Prolog variables can't be changed, but backtracking undoes any binding. Erlang variables cannot be changed after they are bound, and cannot be used before they are bound. Prolog variables can be bound to other variables, in which case they "share" the binding (binding one also binds the other).

11. In three or four sentences, describe the most interesting or unexpected thing you encountered in doing the labs.
12. In 200–300 words, describe what you plan to do in your independent project. State which language you have chosen.