The Puzzle

The puzzle will be summarised here.

This file is referenced in the DIV1 section of the html file.

Abstraction

Some tips and pointers will be included here to help with the process of extracting key information, spotting patterns and generalising the patterns into re-useable algorithms.

Writing your Thinking

Typically students are asked to take 5 minutes to think about how they tackled this problem.

- **Did you use pen and paper to help visualise possible solutions?**
- Did you break it down to a smaller problem to verify your method?
- **Did you look for patterns in order to make it easier to solve?**
- Did you try different scenarios to verify your solution worked?

Using Think-Pair-Share-Square (TPSS), go through how you and your partner were thinking about how to solve the problem.

Pseudo-Code

```
# This is a comment : pseudocode just needs to be consistent and clear.
# Break up your algorithm into smaller sections.
# _____________________________________________________________
# Initialisation
ask the user for data;

# Underline pseudocode that translates to specific code in the
programming language e.g while, if .. else, mod, etc…
remaining Seconds = user's number of Seconds MOD 3600;

# Use {open and close brackets} to signify a chunk of code or a specific
task.

For i = 1 to 3 {
        Make a forward list of [B,C,D] ;
}

If length of chaliceList < 101 {
        Append backward list;
        Append forward list;
}

#report back to the user as part of your User Interface (UI)
output the total number of seconds and the conversion to HMS
```

An extension to your program

Often, when you have working and functional code, it can be much easier to extend the program to do more.

Perhaps use one program's output to verify the correct functionality of another?

Always be thinking of ways to improve your program.