Q1

```
write "factorial: " ;
read n ; i := 1 ;
while ( n > 0 ) do{
i := i * n;
n := n - 1
};
write i;
write "\n"
```

Q2

1.push the final result of AExp in to the stack directly. 2. The stack estimation can be improved.

Q3

Byte code is mainly for platform independent and needs a virtual machine whereas assembly code is human readable machine code that directly run by CPU.
Byte code is not machine/hardware specific but assembly code is hardware/machine specific.
Java assembly creates class files that have concrete addresses to jump. While java byte code creates j files where they have symbols to make jumps.
The j-files have symbols for places where to jump, while class files have this resolved to concrete addresses (or relative jumps). That is what the assembler has to generate.

Q4

iseven & isodd are both tail-recursive
Stack safe: A tail recursive function, which has a recursive call in stack position is stack-safe. But so is a function that doesn't contain any recursive call at all.
iseven & isodd look superficially stack-safe, but because they call each other and scala cannot optimise it, they are not stack-safe.

Q5

lazy evaluation only evaluates expressions when they are needed and if they are needed twice, the results will be re-used.
Eager evaluation immediately evaluates expressions, for example if they are arguments to function calls or allocated to variables.