1. What are some possible reasons for a programmer to split code into multiple files? Select all possible reasons from below.

   (a) Overall code-base is too large for understandability

   (b) Code reuse in the form of libraries

   (c) Incremental compilation on change to a portion of the code-base

   (d) Reducing the final size of the executable

   (e) Making the program run faster

   Answer: (a), (b), (c)

2. Given two files `main.s` and `aux.s`, answert the following questions:

   `main.s`

```
.data Y
main:
...
jal F # I1
...
lw $s0, X # I2
...
G:
...
lw $s1, Y # I3
```

   `aux.s`

```
.data X
F:
...
jal G # I4
...
lw $s2, X # I5
...
...
...
sw Y, $s3 # I6
```

- What is going to be the contents of the relocation table of `main.o`?
  (a) Symbols `main, G, Y`
  (b) Symbols `F, X`
  (c) Machine code for instructions `I1, I2, I3`
  (d) Line numbers (or locations) of `I1, I2, I3`
  (e) Line numbers (or locations) of `I4, I5, I6`

  Answer: (d)
- What is going to be the contents of the symbol table of `main.o`?
  (a) Symbols `main, G, Y`
  (b) Symbols `main, F, G, X, Y`
  (c) Symbols `F, X`
  (d) Line numbers (or locations) of `I1, I2, I3`
  (e) Line numbers (or locations) of instructions `I1` to `I6`

  Answer: (b)
- What is going to be the contents of the relocation table of `aux.o`?
  (a) Symbols `main, G, X`
  (b) Symbols `F, Y`
  (c) Symbols `main, F, G, X, Y`
  (d) Line numbers (or locations) of `I4, I5, I6`
  (e) Line numbers (or locations) of instructions `I1` to `I6`

  Answer: (d)
- What is going to be the contents of the symbol table of `aux.o`?
  (a) Symbols `G, X`
  (b) Symbols `F, G, X, Y`
  (c) Symbols `F, Y`
  (d) Line numbers (or locations) of `I4, I5, I6`
  (e) Line numbers (or locations) of instructions `I1` to `I6`

  Answer: (b)

3. Which entity implements the loader functionality?

   (a) The MIPS Hardware
   (b) The digital circuit

(c) The compiler

(d) The Operating System

(e) The program itself

Answer: (d)

4. Given that the Operating System itself is a program, who loads the OS?

   (a) The hardware implementer

   (b) The OS itself

   (c) The boot loader

   (d) The hard disk

   Answer: (c)

5. Why is it useful to support dynamic linking? Select all possible reasons from below.

   (a) The `.exe` file size would be small with dynamic linking

   (b) The same dynamic library could be shared in memory across multiple processes

   (c) Only the libraries necessary to run the program would be loaded onto memory

   (d) The dynamically linked library could be updated/enhanced long after exe file generation

   (e) The program will run faster with dynamic linking

   Answer: (a), (b), (c), (d)

6. How can we go about implementing dynamic linking?

   Answer:

   - **Rewrite indirect jump address**
   - **Use jump table of function pointers**

   Note: `jalr` instruction needed to support function pointers