Teórica

Vamos fazer esta resposta só por tópicos muito breves.
A partir daqui, o recomendado será elaborar mais cada ponto e dar até exemplos, se possível.

Começar com uma breve descrição do enunciado e do que fizeram no projeto:

Antes de SO:

- O programa não teria concorrência, tendo impactos profundos na performance;
- O programa estaria dependente do input de users, perdendo muito tempo nisto e causando enormes tempos de espera para utilizadores na fila;
- Programa vulnerável a crashes que fariam perder o servidor e os dados;
- Programa usaria sistema de menus que causaria ainda mais atrasos na performance;
- O código do cliente e do servidor iria pertencer ao mesmo programa.

Após SO:

- O programa usaria concorrência, garantindo maior performance;
- O input de users e sistema de menus seria inexistente. Em vez disso, cada user teria um programa cliente que iria dar input ao servidor, ficando o servidor responsável pela sua carga e o cliente apenas teria de enviar os dados;
- Em caso de crash, o programa do projeto iria reiniciar a parte que crashou, garantindo a persistência de dados através da criação de logs.

Memória virtual:

- Explicar no que consiste a memória virtual (essencialmente dizer que cada programa assume que tem a memória toda disponível. A partir daí falar da tabela de páginas e explicar como esses dados são partilhados entre programas);

- Contextualizar em relação ao trabalho. Depende um pouco da alternativa de cada grupo. Provavelmente, o ponto mais importante seria que dizer que quantidade de processos-filho do servidor seria muito grande. Se cada um tivesse a sua memória exclusiva e não fosse partilhada (sem memória virtual), não restaria muita memória para o resto do computador. Como exemplo, o mais óbvio e importante seria a quantidade de bibliotecas que cada processo ia ter na sua memória. Estas seriam iguais para todos e podiam ser facilmente partilhadas. Explicar que é isso que a memória virtual faz através da paginação (mais uma vez, contextualizar).