1) A saída do algoritmo e uma sequência de números primos determinada pelo código. Colocando o mesmo no console do navegador, a resposta é a seguinte:

4603|5021|5477|5869
4621|5023|5479|5879
4637|5039|5483|5881
4639|5051|5501|5897
4643|5059|5503|5903
4649|5077|5507|5923
4651|5081|5519|5927
4657|5087|5521|5939
4663|5099|5527|5953
4673|5101|5531|5981
4679|5107|5557|5987
4691|5113|5563|6007
4703|5119|5569|6011
4721|5147|5573|6029
4723|5153|5581|6037
4729|5167|5591|6043
4733|5171|5623|6047
4751|5179|5639|6053
4759|5189|5641|6067
4783|5197|5647|6073
4787|5209|5651|6079
4789|5227|5653|6089
4793|5231|5657|6091
4799|5233|5659|6101
4801|5237|5669|6113
4813|5261|5683|6121
4817|5273|5689|6131
4831|5279|5693|6133
6143|6577|7001|7507
6151|6581|7013|7517
6163|6599|7019|7523
6173|6607|7027|7529
6197|6619|7039|7537
6199|6637|7043|7541
6203|6653|7057|7547
6211|6659|7069|7549
6217|6661|7079|7559
6221|6673|7103|7561
6229|6679|7109|7573
6247|6689|7121|7577
6257|6691|7127|7583
6263|6701|7129|7589
6269|6703|7151|7591
6271|6709|7159|7603
6277|6719|7177|7607

6287|6733|7187|7621
6299|6737|7193|7639
6301|6761|7207|7643
6311|6763|7211|7649
6317|6779|7213|7669
6323|6781|7219|7673
6329|6791|7229|7681
6337|6793|7237|7687
6343|6803|7243|7691
6353|6823|7247|7699
6359|6827|7253|7703
6361|6829|7283|7717
6367|6833|7297|7723
6373|6841|7307|7727
6379|6857|7309|7741
6389|6863|7321|7753
6397|6869|7331|7757
6421|6871|7333|7759
6427|6883|7349|7789
6449|6899|7351|7793
6451|6907|7369|7817
6469|6911|7393|7823
6473|6917|7411|7829
6481|6947|7417|7841
6491|6949|7433|7853
6521|6959|7451|7867
6529|6961|7457|7873
6547|6967|7459|7877
6551|6971|7477|7879
6553|6977|7481|7883
6563|6983|7487|7901
6569|6991|7489|7907
6571|6997|7499|7919

2) Apesar do código atingir seu propósito, ele poderia ser melhorado em termos de entendimento e leitura, utilizando constantes nomeadas de acordo com sua função, não de forma aleatória, também declarando variáveis mais próximas de seu uso. Além disso, algumas variáveis não utilizadas podem ser removidas para que o código fique mais limpo.

3) resposta no repositório do git dentro do arquivo main.js

4) No express.js os middlewares podem ser utilizados para implementar lógicas e autenticação, manipulação de erros, logs e análise do body da solicitação. Criando esses middlewares, é possível aplicar essas funções em rotas de todo o aplicativo. A vantagem é que isso simplifica as rotas individuais e também promove a pratica do

clean code, além de centralizar todo o comportamento do app em um arquivo, o que facilita a manutenção do mesmo.

5) As vantagens do ORM consistem em permitir que os devs consigam trabalhar com objetos ao invés de ter que consultar diretamente no SQL. Além disso, o ORM facilita a manipulação da comunicação entre entidades, simplificando consultas que envolvem várias tabelas.
Por outro lado, o uso direto de SQL permite operações mais otimizadas e personalidades pela flexibilidade da linguagem.

   Já no KNEX, uma das principais vantagens é que ele oferece total controle sobre as consultas geradas em SQL, permitindo a manipulação direta na DB. Também o fato de que um query builder é muito mais parecido com o SQL puro deixa o trabalho do desenvolvedor muito mais tranquilo.
Minha maior crítica em relação a qualquer construtor de consultas é que ele adiciona um nível desnecessário de abstração sobre o que já é uma linguagem projetada especificamente para criar, ler, atualizar e excluir dados.

6) **SQL Query**

```
SELECT DISTINCT j1.nome AS jogador1, j2.nome AS jogador2
FROM partidas p
JOIN jogador j1 ON p.jogador1_id = j1.id
JOIN jogador j2 ON p.jogador2_id = j2.id
WHERE (p.pontos_jogador1 + p.pontos_jogador2) > 30 AND p.duracao > 90
GROUP BY j1.nome, j2.nome
HAVING COUNT(DISTINCT p.id) > 2;
```

   print do teste: