Emscripten Development Environment Setup for Kali Linux

Install Prerequisites:

1:

EMCC uses a lot of python scripts in the background. Consequently, you need to install python3 if it's not installed already.

```
sudo apt update
sudo apt-get install python3
```

Install other packages that may be needed

```
sudo apt-get install node, wget, cmake
```

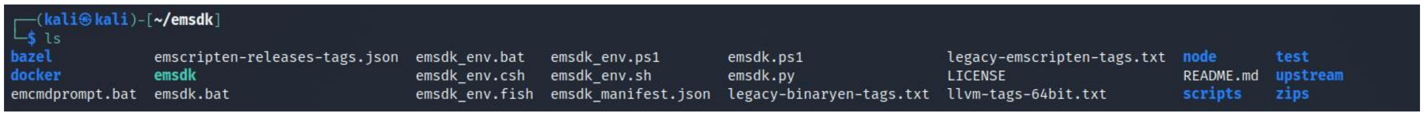
To install VScode IDE

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg –dearmor
> packages.microsoft.gpg
$ sudo install -o root -g root -m 644 packages.microsoft.gpg
/etc/apt/trusted.gpg.d/
$ sudo sh -c 'echo "deb [arch=amd64 signed-
by=/etc/apt/trusted.gpg.d/packages.microsoft.gpg]
https://packages.microsoft.com/repos/vscode stable main" >
/etc/apt/sources.list.d/vscode.list'
sudo apt install code
```

EMSCRIPTEN Installation

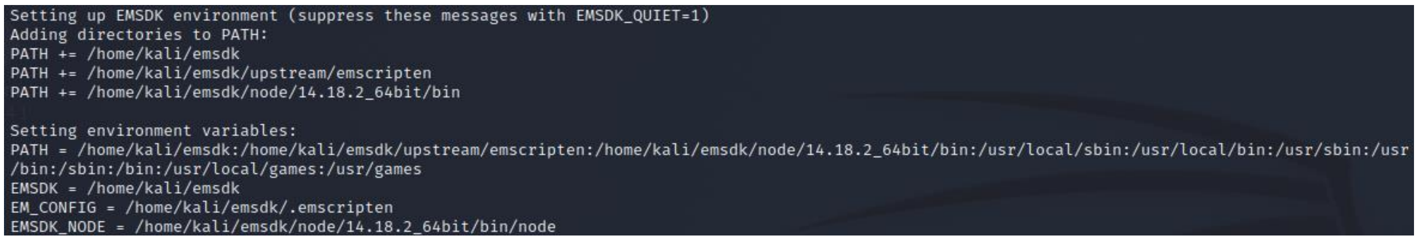
To install emscripten sdk

```
git clone https://github.com/emscripten-core/emsdk.git
cd emsdk
ls emdsk
```



```
┌──(kali㉿kali)-[~/emsdk]
└─$ ls
bazel          emscripten-releases-tags.json  emsdk_env.bat   emsdk_env.ps1        emsdk.ps1                 legacy-emscripten-tags.txt  node       test
docker         emsdk                          emsdk_env.csh   emsdk_env.sh         emsdk.py                  LICENSE                     README.md  upstream
emcmdprompt.bat  emsdk.bat                    emsdk_env.fish  emsdk_manifest.json  legacy-binaryen-tags.txt  llvm-tags-64bit.txt         scripts    zips
```

```
./emsdk install latest
./emsdk activate latest
source ./emsdk_env.sh
```

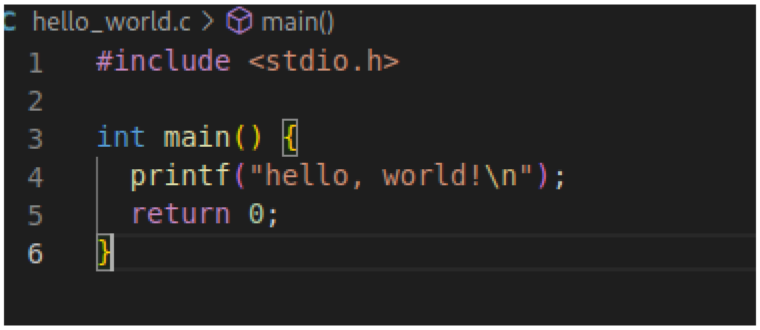


```
Setting up EMSDK environment (suppress these messages with EMSDK_QUIET=1)
Adding directories to PATH:
PATH += /home/kali/emsdk
PATH += /home/kali/emsdk/upstream/emscripten
PATH += /home/kali/emsdk/node/14.18.2_64bit/bin

Setting environment variables:
PATH = /home/kali/emsdk:/home/kali/emsdk/upstream/emscripten:/home/kali/emsdk/node/14.18.2_64bit/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr
/bin:/sbin:/bin:/usr/local/games:/usr/games
EMSDK = /home/kali/emsdk
EM_CONFIG = /home/kali/emsdk/.emscripten
EMSDK_NODE = /home/kali/emsdk/node/14.18.2_64bit/bin/node
```

NOTE:

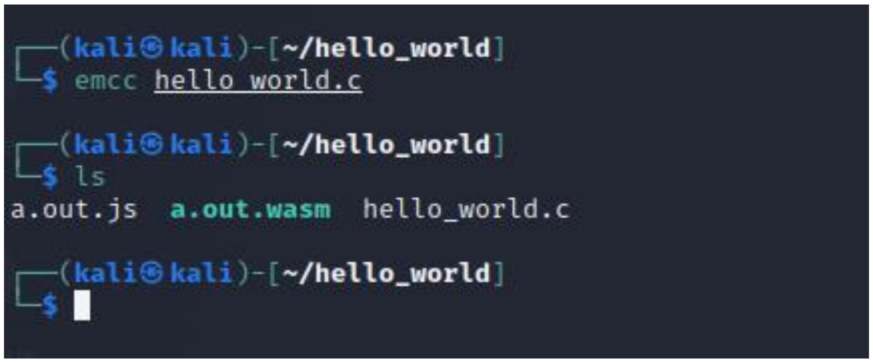
Please not that you need to run source <path to emsdk_env.sh> if you change directory and you need to add emcc command to execution path. To simplify the solution, I add the source command <**source /home/kali/emsdk/emsdk_env.sh>** to my zshrc file.

Verifying Hello World Project



```
hello_world.c > main()
#include <stdio.h>

int main() {
  printf("hello, world!\n");
  return 0;
}
```

- Run *emcc hello_world.c*



```
┌──(kali㉿kali)-[~/hello_world]
└─$ emcc hello_world.c

┌──(kali㉿kali)-[~/hello_world]
└─$ ls
a.out.js  a.out.wasm  hello_world.c

┌──(kali㉿kali)-[~/hello_world]
└─$ 
```

2:

Code Base to Transpile:

https://github.com/epoell/openMP_examples_and_Matrix-Matrix-Multiplication

Code was refactored to suit test case.

matrix.h

```cpp
class Matrix {
    public:
        Matrix();
        double multMatrix(int n);
};

```

matrix.cpp

```cpp
#include "matrix.h"
#include <iostream>
#include <sys/time.h>

using namespace std;

Matrix::Matrix() {

}


double Matrix::multMatrix(int n) {

    double A[n][n], B[n][n], C[n][n];

    // Initialize Matrices

    for(int i = 0; i < n; ++i)
        for(int j = 0; j < n; ++j)
        {
            A[i][j] = (double)rand()/ (double)RAND_MAX;
      B[i][j] = (double)rand()/ (double)RAND_MAX;
      C[i][j] = 0;
        }

    // Matrix multiplication

    int i,j,k;

    struct timeval start, end;

    // start timer.
    gettimeofday(&start, NULL);

    // unsync the I/O of C and C++.
    ios_base::sync_with_stdio(false);


    for(i = 0; i < n; ++i) {
        for(int k = 0; k < n; ++k) {
            for(j = 0; j < n; ++j) {
                        C[i][j] += A[i][k] * B[k][j];
                }
      }
    }

    gettimeofday(&end, NULL);

    double time_taken;

    time_taken = (end.tv_sec - start.tv_sec) * 1e6;
    time_taken = (time_taken + (end.tv_usec -
                              start.tv_usec)) * 1e-6;

    return time_taken;
}

```

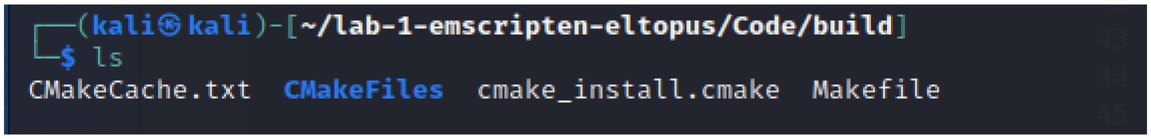
The code takes in an integer value n, creates 3 matrix n * n matrix, populates matrix with random floating-point values and then multiplies the 3 matrices together. The code returns time taken for the computation in secs.

Testing Base Code by compiling with emcc:

- after cloning class GitHub repo, create a ***build*** directory in the repo ***lab-1-emscripten-eltopus/Code*** to hold all cmake configurations.
- create a CMakeLists.txt in ***lab-1-emscripten-eltopus/Code***

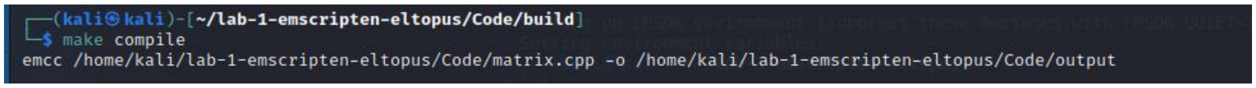
```
cmake_minimum_required(VERSION 3.10)

# set the project name
project(Lab1)

# add the executable
add_executable(matrix matrix.cpp)
```

- run cmake --build ./build from ***lab-1-emscripten-eltopus/Code***
- This created the following files in the build folder



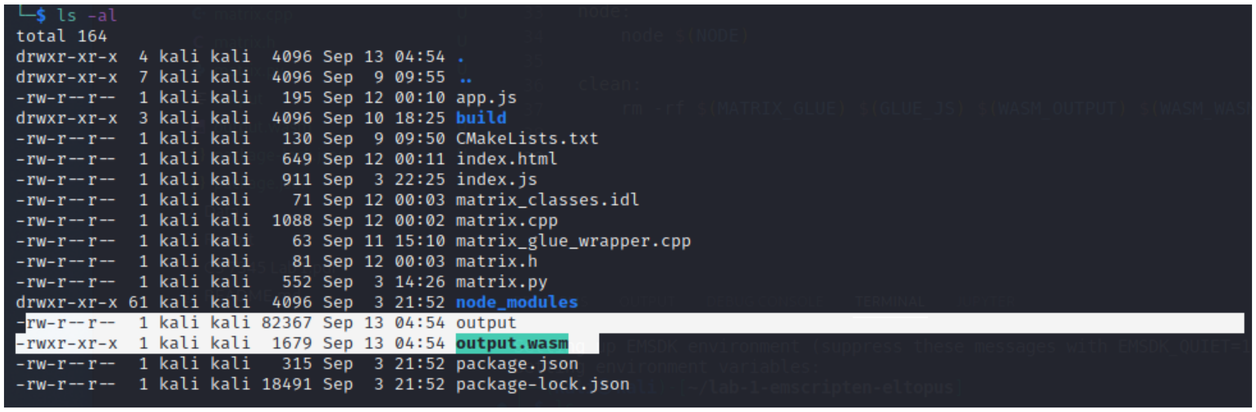
```
┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code/build]
└─$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile
```

- modify Makefile to compile .cpp to web assembly
- run make compile



```
┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code/build]
└─$ make compile
emcc /home/kali/lab-1-emscripten-eltopus/Code/matrix.cpp -o /home/kali/lab-1-emscripten-eltopus/Code/output
```

- Successful compilation should create 2 files in /home/kali/lab-1-emscripten-eltopus/Code/



```
└─$ ls -al
total 164
drwxr-xr-x  4 kali kali  4096 Sep 13 04:54 .
drwxr-xr-x  7 kali kali  4096 Sep  9 09:55 ..
-rw-r--r--  1 kali kali   195 Sep 12 00:10 app.js
drwxr-xr-x  3 kali kali  4096 Sep 10 18:25 build
-rw-r--r--  1 kali kali   130 Sep  9 09:50 CMakeLists.txt
-rw-r--r--  1 kali kali   649 Sep 12 00:11 index.html
-rw-r--r--  1 kali kali   911 Sep  3 22:25 index.js
-rw-r--r--  1 kali kali    71 Sep 12 00:03 matrix_classes.idl
-rw-r--r--  1 kali kali  1088 Sep 12 00:02 matrix.cpp
-rw-r--r--  1 kali kali    63 Sep 11 15:10 matrix_glue_wrapper.cpp
-rw-r--r--  1 kali kali    81 Sep 12 00:03 matrix.h
-rw-r--r--  1 kali kali   552 Sep  3 14:26 matrix.py
drwxr-xr-x 61 kali kali  4096 Sep  3 21:52 node_modules
-rw-r--r--  1 kali kali 82367 Sep 13 04:54 output
-rwxr-xr-x  1 kali kali  1679 Sep 13 04:54 output.wasm
-rw-r--r--  1 kali kali   315 Sep  3 21:52 package.json
-rw-r--r--  1 kali kali 18491 Sep  3 21:52 package-lock.json
```

3:

Build Code base to web assembly

- create matrix_glue_wrapper.cpp

```
#include <stddef.h>
#include "matrix.h"
#include "glue.cpp"
```

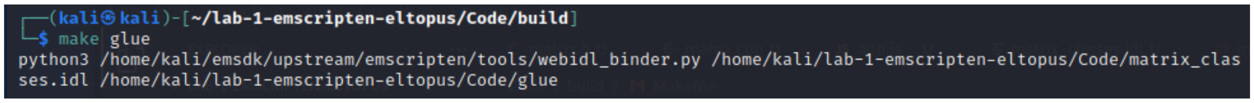
- make the following modifications to ***lab-1-emscripten-eltopus/Code/build/Makefile***
- glue.cpp does not currently exist but will be generated later

```
SHELL = /bin/sh
# The top-level source directory on which CMake was run.
CMAKE_SOURCE_DIR = /home/kali/lab-1-emscripten-eltopus/Code

# The top-level build directory on which CMake was run.
CMAKE_BINARY_DIR = /home/kali/lab-1-emscripten-eltopus/Code/build

MATRIX_CPP = /home/kali/lab-1-emscripten-eltopus/Code/matrix.cpp
WASM_OUTPUT = /home/kali/lab-1-emscripten-eltopus/Code/output
WASM_WASM = /home/kali/lab-1-emscripten-eltopus/Code/output.wasm
MATRIX_GLUE_WRAPPER = /home/kali/lab-1-emscripten-
eltopus/Code/matrix_glue_wrapper.cpp
MATRIX_GLUE = /home/kali/lab-1-emscripten-eltopus/Code/glue.cpp
NODE = /home/kali/lab-1-emscripten-eltopus/Code/app.js
OUTPUT_JS = /home/kali/lab-1-emscripten-eltopus/Code/output.js
GLUE_JS = /home/kali/lab-1-emscripten-eltopus/Code/glue.js
EMSDK_ROOT_FOLDER = /home/kali/emsdk/upstream/emscripten
REPO_CODE_DIR = /home/kali/lab-1-emscripten-eltopus/Code



compile:
	$(SHELL source ./home/kali/emsdk/emsdk_env.sh)
	$(EMSDK_ROOT_FOLDER)/emcc $(MATRIX_CPP) -o $(WASM_OUTPUT)

glue:
	python3 $(EMSDK_ROOT_FOLDER)/upstream/emscripten/tools/webidl_binder.py
$(REPO_CODE_DIR)/matrix_classes.idl $(REPO_CODE_DIR)/glue

build:
	$(EMSDK_ROOT_FOLDER)/emcc $(MATRIX_CPP) $(MATRIX_GLUE_WRAPPER) --post-js
$(GLUE_JS) -o $(OUTPUT_JS) -s EXPORTED_RUNTIME_METHODS=["ccall, cwrap"]

node:
	node $(NODE)

clean:
	rm -rf $(MATRIX_GLUE) $(GLUE_JS) $(WASM_OUTPUT) $(WASM_WASM) $(OUTPUT_JS)
```

- create matrix_classes.idl

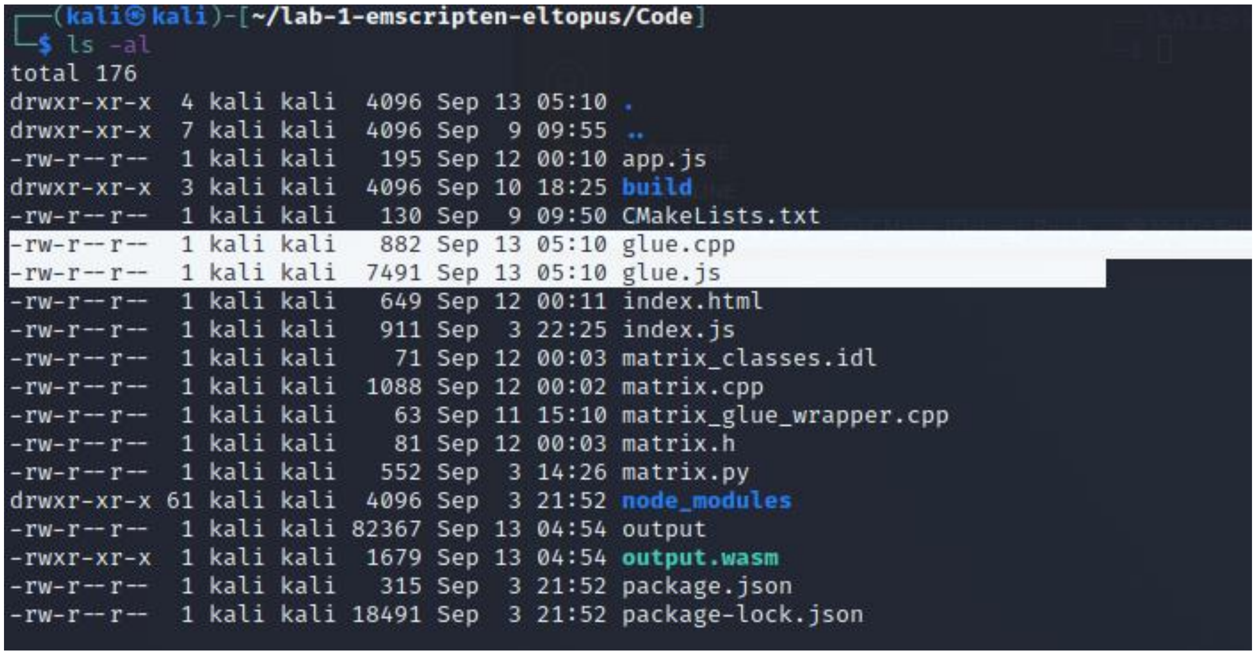
```
interface Matrix {
    void Matrix();
    double multMatrix(long n);
};
```

- run make glue



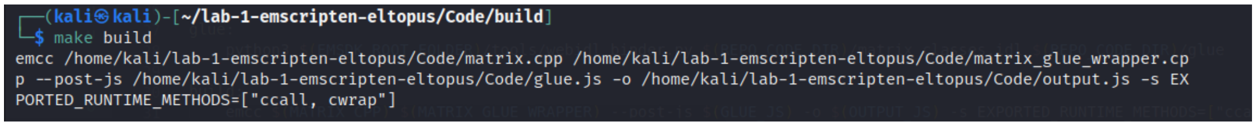
```
┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code/build]
└─$ make glue
python3 /home/kali/emsdk/upstream/emscripten/tools/webidl_binder.py /home/kali/lab-1-emscripten-eltopus/Code/matrix_clas
ses.idl /home/kali/lab-1-emscripten-eltopus/Code/glue
```

- glue.cpp and glue.js files will be generated



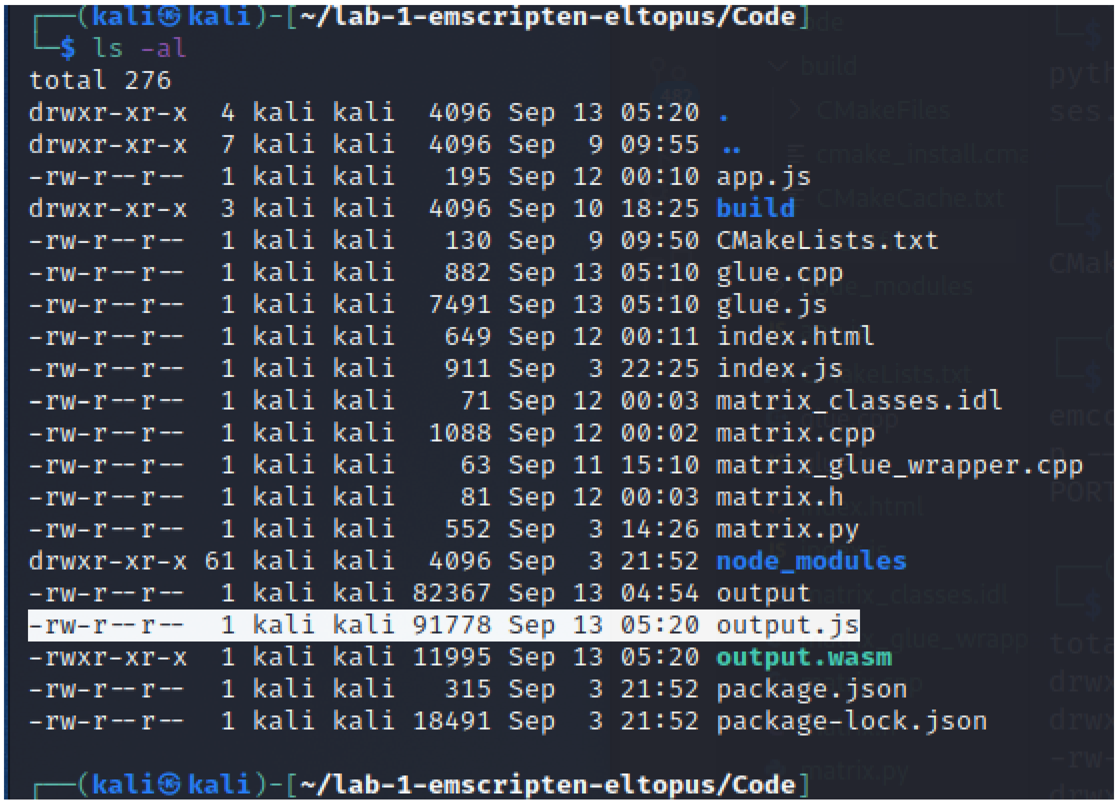
```
┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code]
└─$ ls -al
total 176
drwxr-xr-x  4 kali kali  4096 Sep 13 05:10 .
drwxr-xr-x  7 kali kali  4096 Sep  9 09:55 ..
-rw-r--r--  1 kali kali   195 Sep 12 00:10 app.js
drwxr-xr-x  3 kali kali  4096 Sep 10 18:25 build
-rw-r--r--  1 kali kali   130 Sep  9 09:50 CMakeLists.txt
-rw-r--r--  1 kali kali   882 Sep 13 05:10 glue.cpp
-rw-r--r--  1 kali kali  7491 Sep 13 05:10 glue.js
-rw-r--r--  1 kali kali   649 Sep 12 00:11 index.html
-rw-r--r--  1 kali kali   911 Sep  3 22:25 index.js
-rw-r--r--  1 kali kali    71 Sep 12 00:03 matrix_classes.idl
-rw-r--r--  1 kali kali  1088 Sep 12 00:02 matrix.cpp
-rw-r--r--  1 kali kali    63 Sep 11 15:10 matrix_glue_wrapper.cpp
-rw-r--r--  1 kali kali    81 Sep 12 00:03 matrix.h
-rw-r--r--  1 kali kali   552 Sep  3 14:26 matrix.py
drwxr-xr-x 61 kali kali  4096 Sep  3 21:52 node_modules
-rw-r--r--  1 kali kali 82367 Sep 13 04:54 output
-rwxr-xr-x  1 kali kali  1679 Sep 13 04:54 output.wasm
-rw-r--r--  1 kali kali   315 Sep  3 21:52 package.json
-rw-r--r--  1 kali kali 18491 Sep  3 21:52 package-lock.json
```

- run glue build



```
┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code/build]
└─$ make build
emcc /home/kali/lab-1-emscripten-eltopus/Code/matrix.cpp /home/kali/lab-1-emscripten-eltopus/Code/matrix_glue_wrapper.cp
p --post-js /home/kali/lab-1-emscripten-eltopus/Code/glue.js -o /home/kali/lab-1-emscripten-eltopus/Code/output.js -s EX
PORTED_RUNTIME_METHODS=["ccall, cwrap"]
```

- output.js will be generated.



```
┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code]
└─$ ls -al
total 276
drwxr-xr-x  4 kali kali  4096 Sep 13 05:20 .
drwxr-xr-x  7 kali kali  4096 Sep  9 09:55 ..
-rw-r--r--  1 kali kali   195 Sep 12 00:10 app.js
drwxr-xr-x  3 kali kali  4096 Sep 10 18:25 build
-rw-r--r--  1 kali kali   130 Sep  9 09:50 CMakeLists.txt
-rw-r--r--  1 kali kali   882 Sep 13 05:10 glue.cpp
-rw-r--r--  1 kali kali  7491 Sep 13 05:10 glue.js
-rw-r--r--  1 kali kali   649 Sep 12 00:11 index.html
-rw-r--r--  1 kali kali   911 Sep  3 22:25 index.js
-rw-r--r--  1 kali kali    71 Sep 12 00:03 matrix_classes.idl
-rw-r--r--  1 kali kali  1088 Sep 12 00:02 matrix.cpp
-rw-r--r--  1 kali kali    63 Sep 11 15:10 matrix_glue_wrapper.cpp
-rw-r--r--  1 kali kali    81 Sep 12 00:03 matrix.h
-rw-r--r--  1 kali kali   552 Sep  3 14:26 matrix.py
drwxr-xr-x 61 kali kali  4096 Sep  3 21:52 node_modules
-rw-r--r--  1 kali kali 82367 Sep 13 04:54 output
-rw-r--r--  1 kali kali 91778 Sep 13 05:20 output.js
-rwxr-xr-x  1 kali kali 11995 Sep 13 05:20 output.wasm
-rw-r--r--  1 kali kali   315 Sep  3 21:52 package.json
-rw-r--r--  1 kali kali 18491 Sep  3 21:52 package-lock.json

┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code]
```

Prepare Testing Html Output page with NodeJS server

- run npm init
- follow cli instructions
- run npm install express cors body-parser

NOTE

For this experiment, compiled JavaScript is compared with native JavaScript. Compiled JavaScript is referred to as WASM code.

- Rewrite C++ matrix.cpp to JavaScript matrix.js

```
class JSMatrix {

    multMatrix(n){

        let matrixA = [];
        let matrixB = [];
        let matrixC = [];

        for (let i = 0; i < n; i++){
            matrixA.push([Math.random().toFixed(5), Math.random().toFixed(5)])
            matrixB.push([Math.random().toFixed(5), Math.random().toFixed(5)])
            matrixC.push([Math.random().toFixed(5), Math.random().toFixed(5)])
        }
        const t0 = performance.now();
        this.multiplyMatrices(matrixA, matrixB);
        const t1 = performance.now();


        return (t1 - t0) // miliseconds

    }

    multiplyMatrices(m1, m2) {
        var result = [];
        for (var i = 0; i < m1.length; i++) {
            result[i] = [];
            for (var j = 0; j < m2[0].length; j++) {
                var sum = 0;
                for (var k = 0; k < m1[0].length; k++) {
                    sum += m1[i][k] * m2[k][j];
                }
                result[i][j] = sum;
            }
        }
        return result;
    }
}

```

- Minify and uglify matrix.js to matrix_mini.js
- create index.htm

•

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Advanced Application Programming Lab 1</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container">
      <h1>Matrix Multiplication Completion Time</h1>
      <form onsubmit="runTests();return false">
        <div class="form-group">
          <label for="firstName">Matrix Size</label>
          <input type="text" class="form-control" id="size" placeholder="Enter matrix size" name="matrix_size">
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
      </form>
      <p id="tk"> <font size="5"> WASM code took 0.0 miliseconds</font></p>
  </body>
  <script  type="text/javascript" src="./output.js"> </script>
  <script  type="text/javascript" src="./matrix_mini.js"> </script>
  <script  type="text/javascript" src="./run.js"> </script>
</html>

```

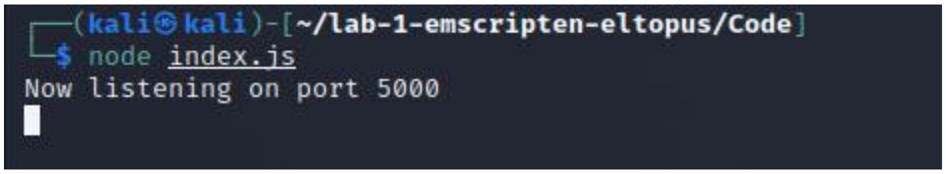
- create run.js file

```
function runTests() {
    const l = document.getElementById("size").value;
    var matrix = new Module.Matrix();
    var jsmatrix = new JSMatrix();

    let total = 0.0;
    let jstotal = 0.0;
    for (x = 0; x < 30; x++){
        const time_taken = matrix.multMatrix(l);
        const js_time_taken = jsmatrix.multMatrix(l);
        total += time_taken;
        jstotal += js_time_taken;
        console.log("WASM Time taken for run #: " + x + ": " + time_taken.toFixed(4) + " milisecs");
        console.log("JS Time taken for run #: " + x + ": " + js_time_taken.toFixed(4) + " milisecs");
    }
    const avg = (total / l).toFixed(4)
    const js_avg = (jstotal/ l).toFixed(4)
    console.log("WASM Average Time taken for " + l + " runs is: " + avg + " milisecs");
    console.log("JavaScript Average Time taken for " + l + " runs is: " + js_avg + " milisecs");
    let input = "<font size=5>WASM Code Average time taken: " + avg + " milisecs" +
    "<br> Java Script Average time taken: " + js_avg + " milisecs" +
    "<br> WASM Code is: " + (Math.abs(js_avg - avg).toFixed(4)) + " milisecs faster than Javascript Code" + "</font><br/>";
    document.getElementById("tk").innerHTML = input;
}

```

- create index.js file

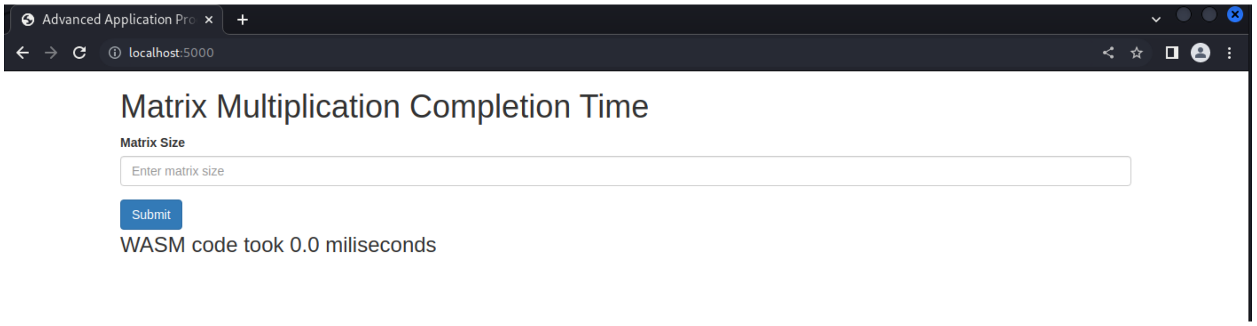
```javascript
const express = require('express'); //Import the express dependency
const cors = require('cors');
const path = require('path');
const bodyParser = require('body-parser')

const app = express();              //Instantiate an express app, the main work horse of this server
const port = 5000;                  //Save the port number where your server will be listening

var urlencodedParser = bodyParser.urlencoded({ extended: false })

app.use(cors());
app.use('/', express.static('.'))

app.get('/', (req, res) => {        //get requests to the root ("/") will route here
    res.sendFile(path.join(__dirname, '/index.html'));    //server responds by sending the index.html file to the client's browser

});

// app.post('/', urlencodedParser, (req, res) => {
//     console.log('Got body:', req.body);
//     res.sendStatus(200);
// });

app.listen(port, () => {            //server starts listening for any attempts from a client to connect at port: {port}
    console.log(`Now listening on port ${port}`);
});

```

- run node index.js



```
┌──(kali㉿kali)-[~/lab-1-emscripten-eltopus/Code]
└─$ node index.js
Now listening on port 5000
```

- launch Google Chrome browser and navigate to localhost:5000



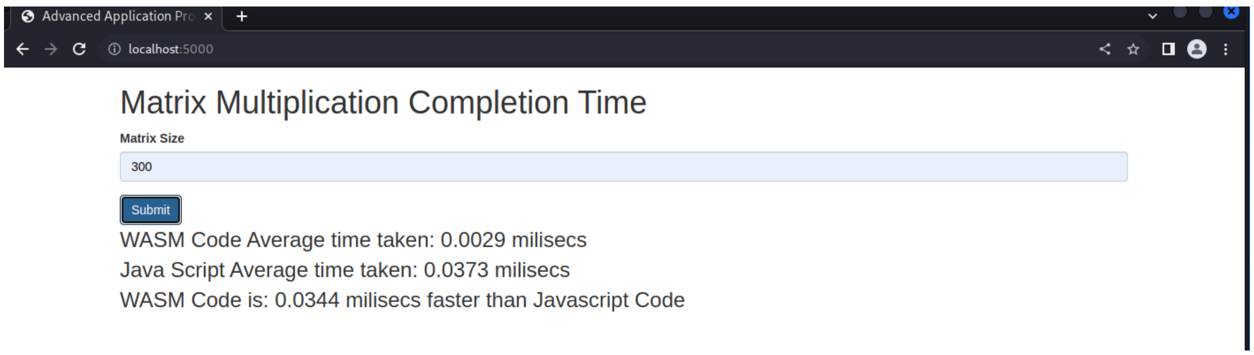
Matrix Multiplication Completion Time

Matrix Size

Enter matrix size

Submit

WASM code took 0.0 miliseconds



Matrix Multiplication Completion Time

Matrix Size

300

Submit

WASM Code Average time taken: 0.0029 milisecs

Java Script Average time taken: 0.0373 milisecs

WASM Code is: 0.0344 milisecs faster than Javascript Code

Conclusion:

Run.js runs 30 iterations of WASM and JavaScript Code and compares the average time taken of both code base. JavaScript code was minified and uglified to mimic the state of WASM code. Still, WASM code was faster than JavaScript code for all iterations.