Task 1

Allow the user to enter whether he/she wants to use a CPU or GPU device. Based on the user's selection, search the system for all CPU or GPU devices. (Note that some systems have multiple CPUs and GPUs).

```
bool correctInput = false;
while (!correctInput) {
    std::cout << "Select a device type (CPU or GPU): " << std::endl;
    std::cout << "---------------------------------------" << std::endl;
    std::cout << "0: CPU" << std::endl;
    std::cout << "1: GPU" << std::endl;
    std::cout << "---------------------------------------" << std::endl;
    std::cin >> inputString;
    std::transform(inputString.begin(), inputString.end(), inputString.begin(), ::toupper);
    if (inputString == "0") {
        correctInput = true;
        //Assignment 1 Task 1.1: set Device type
        deviceChosen = selectDeviceType("CPU");
    }else if (inputString == "1") {
        correctInput = true;
        //Assignment 1 Task 1.1: set Device type
        deviceChosen = selectDeviceType("GPU");
    }
    else{
        std::cout << " Please enter a valid input (CPU/GPU)" << std::endl;
    }
}
```

Get user input 0/1, to decide the use of CPU, or GPU devices

```
cl_device_type chosenDeviceType(std::string inputString) {
    cl_device_type chosenDeviceType;
    std::transform(inputString.begin(), inputString.end(), inputString.begin(), ::toupper);
    if (inputString == "GPU" || inputString == "CPU") {
        if (inputString == "CPU") {
            chosenDeviceType = CL_DEVICE_TYPE_CPU;
        }
        else if (inputString == "GPU") {
            chosenDeviceType = CL_DEVICE_TYPE_GPU;
        }
    }
    return chosenDeviceType;
}
```

Based on input from user, pass in the device type to chosenDeviceType variable

```
Select a device type (CPU or GPU):
---------------------------------------
0: CPU
1: GPU
---------------------------------------
1
```

If the chosen device type does not have a available device, error message pop up and the program automatic exit

```
Select a device type (CPU or GPU):
-------------------------------------
0: CPU
1: GPU
-------------------------------------
0

No available device
Exiting the program...

press a key to quit...
```

When detected available device, program will display all available devices to the platform via for loop

```
//Available OpenCL platforms on the system.
cl::Platform::get(&platforms);
//Looping thru the vectors storing the platforms.
for (i = 0; i < platforms.size(); i++) {
    // get all devices available to the platform
    // platforms[i].getDevices(deviceChoice, &devices);
    platforms[i].getDevices(CL_DEVICE_TYPE_ALL, &devices);
    // store the avail devices for the platform
    platformDevices.push_back(devices);
    //Looping thru the devices
    for (j = 0; j < devices.size(); j++) {
        devices[j].getInfo(CL_DEVICE_TYPE, &deviceType);
        if (deviceType == deviceChosen) {
            //Get Platform name
            platformName = platforms[i].getInfo(CL_PLATFORM_NAME, &outputString);
            std::cout << "Device " << j + 1 << std::endl;
            std::cout << "Platform - " << outputString << std::endl;

            //Get Device Type
            // get and output device type
            if (deviceType == CL_DEVICE_TYPE_CPU)
                std::cout << "Device type - " << "CPU" << std::endl;
            else if (deviceType == CL_DEVICE_TYPE_GPU)
                std::cout << "Device type - " << "GPU" << std::endl;
```

```
//Get Device Name
deviceName = devices[j].getInfo(CL_DEVICE_NAME, &outputString);
std::cout << "Device name - " << outputString << std::endl;
//Get Max compute units
devices[j].getInfo(CL_DEVICE_MAX_COMPUTE_UNITS, &outputInt);
std::cout << "Number of compute unites - " << outputInt << std::endl;
//Get Max work group size
devices[j].getInfo(CL_DEVICE_MAX_WORK_GROUP_SIZE, &outputInt);
std::cout << "Max work-item sizes - " << outputInt << std::endl;
//Get Max work item dimensions
devices[j].getInfo(CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS, &outputInt);
std::cout << "Max dimensions - " << outputInt << std::endl;
//Get Max work item sizes
devices[j].getInfo(CL_DEVICE_MAX_WORK_ITEM_SIZES, &workItem);
std::cout << "Max work-item sizes - [" << workItem[0] << ", " << workItem[1] << ", " << workItem[2] << "]" << std::endl;
//Get dev global memory size
globalMemorySize = devices[j].getInfo<CL_DEVICE_GLOBAL_MEM_SIZE>();
std::cout << "Global memory size - " << globalMemorySize << " KB" << std::endl;
//Get dev Local memory size
localMemorySize = devices[j].getInfo<CL_DEVICE_LOCAL_MEM_SIZE>();
std::cout << "Local memory size - " << localMemorySize << " KB" << std::endl;
platformsID.push_back(i);
devicesID.push_back(j);
```

Output of all available devices and their specifics

```
Select a device type (CPU or GPU):
-------------------------------------
0: CPU
1: GPU
-------------------------------------
1
Device 1
Platform - NVIDIA CUDA
Device type - GPU
Device name - NVIDIA GeForce RTX 2060
Number of compute unites - 30
Max work-item sizes - 1024
Max dimensions - 3
Max work-item sizes - [1024, 1024, 64]
Global memory size - 2147024896 KB
Local memory size - 49152 KB

--------------------
Device options:
--------------------
1: Platform - NVIDIA CUDA, NVIDIA GeForce RTX 2060
--------------------
Select a device:
```

Based on the devices available, allow the user to select one device. Create a context using that device, and a command queue.

Passing in variables to createContext() method

```
//fn to create context on device
// create context
cl::Platform::get(&platforms);
if (platformsID.size() != 0)
    createContext(platforms, platformsID, devices, devicesID, contexts, contextDevice, queue);
else
    quit_program("No available device");
```

Display the available devices for user to select.

```
void createContext(std::vector<cl::Platform>& platforms, std::vector<int>& platformsID, std::vector<cl::Device>& devices,
    std::vector<int>& devicesID, cl::Context& context, std::vector<cl::Device>& contextDevice, cl::CommandQueue& queue) {
    bool correctInput = false;
    std::string getInput;
    std::cout << "--------------------" << std::endl;
    std::cout << "Device options: " << std::endl;
    std::cout << "--------------------" << std::endl;

    for (int i = 0; i < platformsID.size(); i++) {
        context = cl::Context(devices[devicesID[i]]);
        contextDevice = context.getInfo<CL_CONTEXT_DEVICES>();
        std::cout << i + 1 << ": Platform - " << platforms[platformsID[i]].getInfo<CL_PLATFORM_NAME>()
            << ", " << contextDevice[i].getInfo<CL_DEVICE_NAME>() << "\n--------------------" << std::endl;
    }
```

After user input, create context, and command queue of selected device

```
while (!correctInput) {
    std::cout << "Select a device: ";
    std::cin >> getInput;
    unsigned int num = std::atoi(getInput.c_str());
    if (num <= devicesID.size() && num != 0) {
        platforms[platformsID[num - 1]].getDevices(CL_DEVICE_TYPE_ALL, &devices);
        context = cl::Context(devices[devicesID[num - 1]]);
        contextDevice = context.getInfo<CL_CONTEXT_DEVICES>();
        queue = cl::CommandQueue(context, devices[devicesID[num - 1]]);
        std::cout << "--------------------" << std::endl;
        std::cout << "Program build: Successful " << std::endl;
        std::cout << "--------------------" << std::endl;
        correctInput = true;
    }
    else {
        std::cout << "--------------------" << std::endl;
        std::cout << "Program build: Failure" << std::endl;
        std::cout << "--------------------" << std::endl;
    }
}
```

Output

```
--------------------
Device options:
--------------------
1: Platform - NVIDIA CUDA, NVIDIA GeForce RTX 2060
--------------------
Select a device: 1
--------------------
Program build: Successful
--------------------
```

Read the program source code from the provided “task1.cl” file and build the program. Display whether or not the program built successfully and display the program build log if any(display the build log even if the program built successfully).

```
//loads the contents from the file into the string programString and displays its contents
std::string programString(std::istreambuf_iterator<char>(programFile),
    (std::istreambuf_iterator<char>()));
//create program source from one input string
cl::Program::Sources source(1, std::make_pair(programString.c_str(), programString.length() + 1));
//Create a context for the program obj
cl::Program program(contexts, source);
//Try to build the program. If not, display error.
try {
    program.build(contextDevice);
    //Output build log of context deviceS.
    for (int i = 0; i < contextDevice.size(); i++) {
        //get the cpntext device name
        outputString = contextDevice[i].getInfo<CL_DEVICE_NAME>();
        std::string buildLog =program.getBuildInfo<CL_PROGRAM_BUILD_LOG>(contextDevice[i]);
        //output device name
        std::cout << "Device " << outputString << " build log:" << std::endl;
        //output the build log of the program.
        std::cout << buildLog << std::endl;
    }
}
```

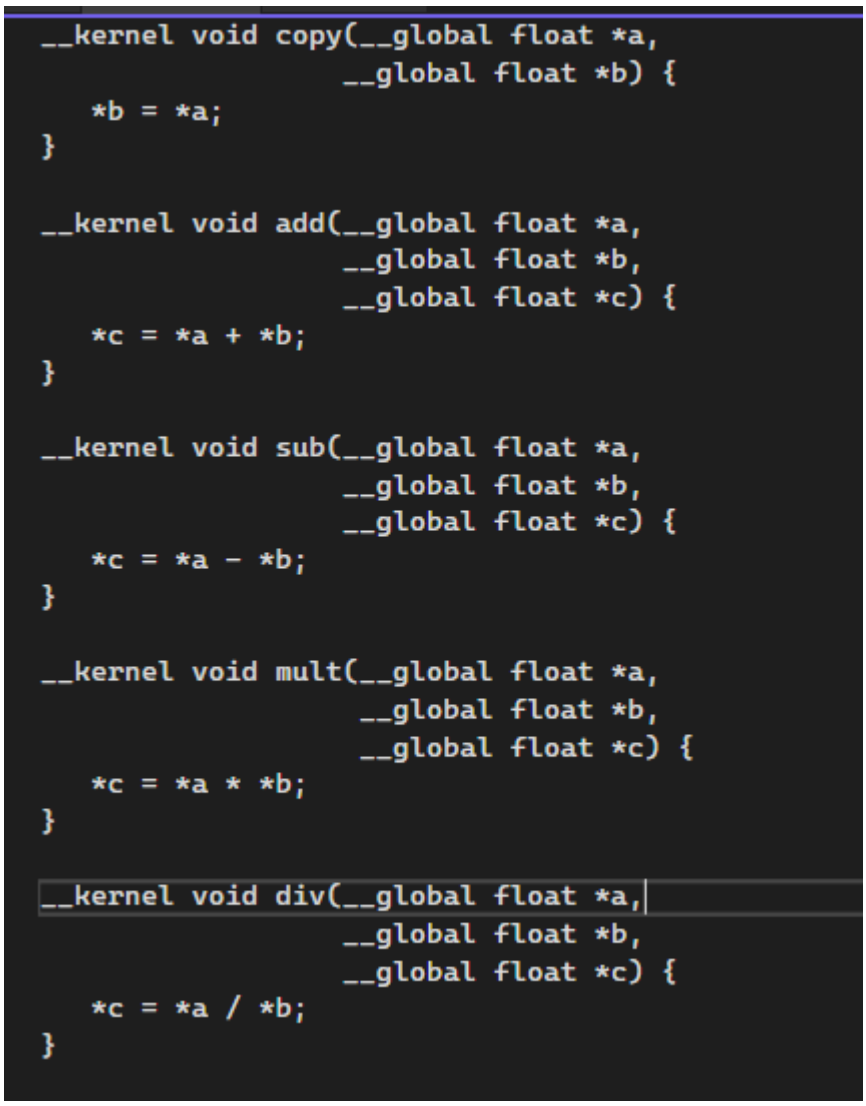
Output

```
---------------------
Device NVIDIA GeForce RTX 2060 build log:


---------------------
```

Find and display the number of kernels in the program. Create kernels from the program and display all the kernel names.

the five kernels from Task1.cl will be used.



```
__kernel void copy(__global float *a,
                   __global float *b) {
    *b = *a;
}

__kernel void add(__global float *a,
                  __global float *b,
                  __global float *c) {
    *c = *a + *b;
}

__kernel void sub(__global float *a,
                  __global float *b,
                  __global float *c) {
    *c = *a - *b;
}

__kernel void mult(__global float *a,
                   __global float *b,
                   __global float *c) {
    *c = *a * *b;
}

__kernel void div(__global float *a,
                  __global float *b,
                  __global float *c) {
    *c = *a / *b;
}
```

Displays total number of kernels, as well as each of the kernel's name

```
    //create kernels in prog
    program.createKernels(&allKernels);
    //output number of kernels
    std::cout << "Number of Kernels - " << allKernels.size() << std::endl;
    for (i = 0; i < allKernels.size(); i++) {
        //get the kernel names
        outputString = allKernels[i].getInfo<CL_KERNEL_FUNCTION_NAME>();
        //output kernel names
        std::cout << "Kernel " << i << " : " << outputString << std::endl;
    }
    std::cout << "--------------------" << std::endl;
}
catch(cl::Error err) {
    handle_error(err);
}
```

Output

```
--------------------
Number of Kernels - 5
Kernel 0 : copy
Kernel 1 : add
Kernel 2 : sub
Kernel 3 : mult
Kernel 4 : div
--------------------
```

Task 2

Create a C++ vector of unsigned chars to store alphabets. Initialise its contents to: a-z and A-Z (i.e. 52 alphabets in total). Create another C++ vector to store 512 unsigned ints. Initialise its contents to: 1-512.

Define and create CHARLENGTH and INTLENGTH and give it 52 and 512 allocated memory

```
#define CHARLENGTH 52
#define INTLENGTH 512
```

Create multiple vectors to store the int, capital and small alphabets, combined alphabets.

```
//Create 2 vector, 1 with unsigned char of 52, another with unsigned int of 512.
std::vector<cl_uchar> alphabets;
std::vector<cl_uchar> alphabetsSmall (CHARLENGTH /2);
std::vector<cl_uchar> alphabetsCap (CHARLENGTH /2);
std::vector<cl_uint> intVector;
```

Fill in the vector with contents (a-z, A-Z, 1-512)

```
//Fill vector with small alphabets a to z
std::iota(alphabetsSmall.begin(), alphabetsSmall.end(), 'a');
//Fill vector with large alphabets A to Z
std::iota(alphabetsCap.begin(), alphabetsCap.end(), 'A');

//Merge the 2 vector (alhpabetsSmall and alphabertsCap) into vector alphabets
alphabets = alphabetsSmall + alphabetsCap;

//Fill the vector with ints
for (i = 0; i < INTLENGTH; i++) {
    intVector.push_back(fillInt);
    fillInt++;
}
```

Give template of concatenation for 2 vectors

```
// template to concatenate 2 vectors.
template <typename T>
std::vector<T> operator+(const std::vector<T>& a, const std::vector<T>& b){
    std::vector<T> ab;
    ab.reserve(a.size() + b.size());                // preallocate memory
    ab.insert(ab.end(), a.begin(), a.end());        // add vector a into ab;
    ab.insert(ab.end(), b.begin(), b.end());        // add vector b into ab;
    return ab;
}
```

Concatenate the two vectors

```
//Merge the 2 vector (alhpabetsSmall and alphabertsCap) into vector alphabets
alphabets = alphabetsSmall + alphabetsCap;
```

Create three OpenCL memory objects (i.e. cl::Buffer objects):

o The first buffer is read-only and initialised with the contents of the alphabet vector.

o The second buffer is write-only and created to store 52 unsigned chars.

o The third buffer is read-and-write and created to store 512 unsigned ints.

Create 3 OpenCL memory buffer objects

```
//Assignment 1 Task 2.2 (Create three openCL mem obj)
cl::Buffer firstBuffer;         //read only and initialised with the contents of alphabet vector
cl::Buffer secondBuffer;        //write-only and created to store 52 unsigned chars.
cl::Buffer thirdBuffer;         //read-and-write and created to store 512 unsigned ints.
```

Create context from chosen device, create buffer for each obj with the correct content assigning, and create a command queue

```
// create a context from device
context = cl::Context(device);

// create buffers
//first buffer read-only, init with contents of the alphabet vector
firstBuffer = cl::Buffer(context, CL_MEM_READ_ONLY | CL_MEM_USE_HOST_PTR, sizeof(cl_uchar) * alphabets.size(), &alphabets[0]);
//second buffer write-only, created to store 52 unsigned char
secondBuffer = cl::Buffer(context, CL_MEM_WRITE_ONLY, sizeof(cl_uchar) * CHARLENGTH);
//third buffer read n write, created to store 512 unsigned char
thirdBuffer = cl::Buffer(context, CL_MEM_READ_WRITE, sizeof(cl_uint) * INTLENGTH);

// create command queue
queue = cl::CommandQueue(context, device);
```

Enqueue two OpenCL commands:

o To copy the contents from the first buffer into the second buffer.

o To write the contents from the vector of 512 integers into the third buffer.

```
//enqueue copy content of first buffer into the second buffer.
queue.enqueueCopyBuffer(firstBuffer, secondBuffer, 0, 0, sizeof(cl_uchar) * CHARLENGTH);
//write content from vector of 512 int to third buffer
queue.enqueueWriteBuffer(thirdBuffer, CL_TRUE, 0, sizeof(cl_uint) * intVector.size(), &intVector[0]);
```

Setup the OpenCL program to allow the user to select one device, create a context and command queue for that device. Then, build the provided “task2.cl” program and create a kernel for “task2”.

All User interface settings is done in common.cpp, with almost exact set up as task 1

```
try {
    // get the number of available OpenCL platforms
    cl::Platform::get(&platforms);
    std::cout << "Number of OpenCL platforms: " << platforms.size() << std::endl;

    // find and store the devices available to each platform
    for (i = 0; i < platforms.size(); i++)
    {
        std::vector<cl::Device> devices;        // available devices

        // get all devices available to the platform
        platforms[i].getDevices(CL_DEVICE_TYPE_ALL, &devices);

        // store available devices for the platform
        platformDevices.push_back(devices);
    }

    // display available platforms and devices
    std::cout << "--------------------" << std::endl;
    std::cout << "Available options:" << std::endl;

    // store options as platform and device indices
    std::vector< std::pair<int, int> > options;
    unsigned int optionCounter = 0; // option counter
```

```
// for all platforms
for (i = 0; i < platforms.size(); i++)
{
    // for all devices per platform
    for (j = 0; j < platformDevices[i].size(); j++)
    {
        // display options
        std::cout << "Option " << optionCounter << ": Platform - ";

        // platform vendor name
        outputString = platforms[i].getInfo<CL_PLATFORM_VENDOR>();
        std::cout << outputString << ", Device - ";

        // device name
        outputString = platformDevices[i][j].getInfo<CL_DEVICE_NAME>();
        std::cout << outputString << std::endl;

        // store option
        options.push_back(std::make_pair(i, j));
        optionCounter++; // increment option counter
    }
}

std::cout << "\n--------------------" << std::endl;
std::cout << "Select a device: ";
```

```
 std::string inputString;
 unsigned int selectedOption;    // option that was selected

 std::getline(std::cin, inputString);
 std::istringstream stringStream(inputString);

 // check whether valid option selected
 // check if input was an integer
 if (stringStream >> selectedOption)
 {
     char c;

     // check if there was anything after the integer
     if (!(stringStream >> c))
     {
         // check if valid option range
         if (selectedOption >= 0 && selectedOption < optionCounter)
         {
             // return the platform and device
             int platformNumber = options[selectedOption].first;
             int deviceNumber = options[selectedOption].second;

             *platfm = platforms[platformNumber];
             *dev = platformDevices[platformNumber][deviceNumber];

             return true;
         }
     }
 }
```

Task2.cl content

```
__kernel void task2(float value,
                    __global unsigned char *copied,
                    __global unsigned int *integers)
{
}
```

Build program

```
// build the program
if (!build_program(&program, &context, "task2.cl"))
{
    // if OpenCL program build error
    quit_program("OpenCL program build error.");
}
```

Actual build_program method() -> common.cpp

```
// builds program from given filename
bool build_program(cl::Program* prog, const cl::Context* ctx, const std::string filename)
{
    // get devices from the context
    std::vector<cl::Device> contextDevices = ctx->getInfo<CL_CONTEXT_DEVICES>();

    // open input file stream to .cl file
    std::ifstream programFile(filename);

    // check whether file was opened
    if (!programFile.is_open())
    {
        std::cout << "File not found." << std::endl;
        return false;
    }
```

```
// create program string and load contents from the file
std::string programString(std::istreambuf_iterator<char>(programFile), (std::istreambuf_iterator<char>()));

// create program source from one input string
cl::Program::Sources source(1, std::make_pair(programString.c_str(), programString.length() + 1));
// create program from source
*prog = cl::Program(*ctx, source);

// try to build program
try {
    // build the program for the devices in the context
    prog->build(contextDevices);

    std::cout << "Program build: Successful" << std::endl;
    std::cout << "--------------------" << std::endl;
}
```

Error catching for build program

```
catch (cl::Error e) {
    // if failed to build program
    if (e.err() == CL_BUILD_PROGRAM_FAILURE)
    {
        // output program build log
        std::cout << e.what() << ": Failed to build program." << std::endl;

        // check build status for all all devices in context
        for (unsigned int i = 0; i < contextDevices.size(); i++)
        {
            // get device's program build status and check for error
            // if build error, output build log
            if (prog->getBuildInfo<CL_PROGRAM_BUILD_STATUS>(contextDevices[i]) == CL_BUILD_ERROR)
            {
                // get device name and build log
                std::string outputString = contextDevices[i].getInfo<CL_DEVICE_NAME>();
                std::string build_log = prog->getBuildInfo<CL_PROGRAM_BUILD_LOG>(contextDevices[i]);

                std::cout << "Device - " << outputString << ", build log:" << std::endl;
                std::cout << build_log << "--------------------" << std::endl;
            }
        }

        return false;
    }
```

Create kernel

```
// create a kernel
kernel = cl::Kernel(program, "task2");
```

• Set kernel arguments for the kernel that was previously created. For the first argument, pass a floating point value of 12.34 to the kernel. For the second and third kernel arguments, set these to the second and third buffers that were previously created. Then, enqueue the kernel using the enqueueTask function.

```
float a = 21.43;
kernel.setArg(0, a);
kernel.setArg(1, secondBuffer);
kernel.setArg(2, thirdBuffer);
queue.enqueueTask(kernel);
```

After returning from the enqueueTask function, read the contents from the three buffers, and display the results on screen.

```
queue.enqueueReadBuffer(secondBuffer, CL_TRUE, 0, sizeof(cl_uchar) * CHARLENGTH, &charOutput[0]);
std::cout << "\nContents of second buffer after kernel execution: " << std::endl;
for (int i = 0; i < CHARLENGTH; i++){
    std::cout << charOutput[i] << " ";
}
queue.enqueueReadBuffer(thirdBuffer, CL_TRUE, 0, sizeof(cl_uint) * INTLENGTH, &intOutput[0]);
std::cout << "\n\nContents of third buffer after kernel execution: ";
for (int i = 0; i < INTLENGTH; i++){
    if ((i % 25) == 0)
    {
        std::cout << "\n";
        std::cout << intOutput[i] << " ";
    }
    else
    {
        std::cout << intOutput[i] << " ";
    }

}
```

Output

User choosing device

```
Number of OpenCL platforms: 1
--------------------
Available options:
Option 0: Platform - NVIDIA Corporation, Device - NVIDIA GeForce RTX 2060

--------------------
Select a device: 0
Program build: Successful
--------------------
Kernel enqueued.
--------------------
```

Display output of the contents of the 2 buffers

```
Contents of second buffer after kernel execution:
a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Contents of third buffer after kernel execution:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225
226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250
251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275
276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325
326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350
351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375
376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425
426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475
476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500
501 502 503 504 505 506 507 508 509 510 511 512
D:\1_Uni\CSCI376\A1\Task2\Debug\Task2.exe (process 4812) exited with code 0.
```

Task 3

Write an OpenCL program that uses a kernel (you will have to write the kernel yourself) to fill in the contents of an array of 1024 numbers in parallel. The program is to prompt the user to enter a number between 1 and 89 (inclusive). The program is to check whether the user entered a valid number, if not the program will quit. If a valid number was entered, enqueue a kernel (using the enqueueNDRangeKernel function) that accepts the number and an array, and fills in the contents of the array using the number (and the work-items' global IDs) as follows:

Define LENGTH to be 1024

```
#define LENGTH 1024
```

Create variables

```
cl::Platform platform;          // device's platform
cl::Device device;              // device used
cl::Context context;            // context for the device
cl::Program program;            // OpenCL program object
cl::Kernel kernel;              // a single kernel object
cl::CommandQueue queue;         // commandqueue for a context and device
unsigned int i;
int inputMultiplier; //store user's input
std::vector<cl_uint> intVector(LENGTH); //host side data obj
cl::Buffer buffer, resultBuffer;    //device side data obj
```

Asking for user input, similar to task 1 and 2

```
//Standard select a device in platform, create context, build program, create buffer, create kernel, set kernel args and create comand queue
try {
    // select an OpenCL device
    if (!select_one_device(&platform, &device))
    {
        // if no device selected
        quit_program("Device not selected.");
    }
```

select_one_device() method in common.cpp

```cpp
bool select_one_device(cl::Platform* platfm, cl::Device* dev)
{
    std::vector<cl::Platform> platforms;    // available platforms
    std::vector< std::vector<cl::Device> > platformDevices; // devices available for each platform
    std::string outputString;               // string for output
    unsigned int i, j;                      // counters

    try {
        // get the number of available OpenCL platforms
        cl::Platform::get(&platforms);
        std::cout << "Number of OpenCL platforms: " << platforms.size() << std::endl;

        // find and store the devices available to each platform
        for (i = 0; i < platforms.size(); i++)
        {
            std::vector<cl::Device> devices;        // available devices

            // get all devices available to the platform
            platforms[i].getDevices(CL_DEVICE_TYPE_ALL, &devices);

            // store available devices for the platform
            platformDevices.push_back(devices);
        }

        // display available platforms and devices
        std::cout << "--------------------" << std::endl;
        std::cout << "Available options:" << std::endl;
```

```cpp
// store options as platform and device indices
std::vector< std::pair<int, int> > options;
unsigned int optionCounter = 0; // option counter

// for all platforms
for (i = 0; i < platforms.size(); i++)
{
    // for all devices per platform
    for (j = 0; j < platformDevices[i].size(); j++)
    {
        // display options
        std::cout << "Option " << optionCounter << ": Platform - ";

        // platform vendor name
        outputString = platforms[i].getInfo<CL_PLATFORM_VENDOR>();
        std::cout << outputString << ", Device - ";

        // device name
        outputString = platformDevices[i][j].getInfo<CL_DEVICE_NAME>();
        std::cout << outputString << std::endl;

        // store option
        options.push_back(std::make_pair(i, j));
        optionCounter++; // increment option counter
    }
}
```

```
std::cout << "\n--------------------" << std::endl;
std::cout << "Select a device: ";

std::string inputString;
unsigned int selectedOption;    // option that was selected

std::getline(std::cin, inputString);
std::istringstream stringStream(inputString);

// check whether valid option selected
// check if input was an integer
if (stringStream >> selectedOption)
{
    char c;

    // check if there was anything after the integer
    if (!(stringStream >> c))
    {
        // check if valid option range
        if (selectedOption >= 0 && selectedOption < optionCounter)
        {
            // return the platform and device
            int platformNumber = options[selectedOption].first;
            int deviceNumber = options[selectedOption].second;

            *platfm = platforms[platformNumber];
            *dev = platformDevices[platformNumber][deviceNumber];

            return true;
        }
    }
}
```

```
    // if invalid option selected
    std::cout << "\n--------------------" << std::endl;
    std::cout << "Invalid option." << std::endl;
}
// catch any OpenCL function errors
catch (cl::Error e) {
    // call function to handle errors
    handle_error(e);
}

return false;
```

Create context as well as build program based on task3.cl

```
// create a context from device
context = cl::Context(device);

// build the program
if (!build_program(&program, &context, "task3.cl"))
{
    // if OpenCL program build error
    quit_program("OpenCL program build error.");
}
```

Task3.cl

```
__kernel void fillArray(int a,   //index 0 inputMultiplier
                        __global int *b) {     //index 1 in/output array

    int i = get_global_id(0); //0 - 1024 work items
    int arry[1024];
    arry[i] = i;
    b[i] = (arry[i] * a) + 1;

}
```

Getting user input of number between 1 -89 inclusive, storing in variable inputMultiplier

```
//Get user's input of multiplier.
std::cout << std::endl;
std::cout << "Please enter a multipler between 1 and 89 (inclusive): ";
std::cin >> inputMultiplier;
if (inputMultiplier < 1 || inputMultiplier > 89) {
    std::cin.clear();
    std::cin.ignore(INT_MAX, '\n');
    quit_program("Please enter a valid integer between 1 and 89. ");
}
else {
    std::cin.clear();
    std::cin.ignore(INT_MAX, '\n');
}
```

Create kernel, command queue, buffer as well as set kernel arguments to variables inputMultiplier and buffer.

```
}

// create a kernel
kernel = cl::Kernel(program, "fillArray");

// create command queue
queue = cl::CommandQueue(context, device);

// create buffers
buffer = cl::Buffer(context, CL_MEM_READ_WRITE | CL_MEM_USE_HOST_PTR, sizeof(cl_uint) * LENGTH, &intVector[0]);

// set kernel arguments
kernel.setArg(0, inputMultiplier);
kernel.setArg(1, buffer);
```

Using enqueueNDRangeKernel, enqueue the kernel (LENGTH 1024)

```
// enqueue kernel for execution 1024
queue.enqueueNDRangeKernel(kernel, cl::NDRange(0), cl::NDRange(LENGTH));

std::cout << "\nKernel enqueued." << std::endl;
std::cout << "--------------------" << std::endl;

// enqueue command to read from device to host memory
queue.enqueueReadBuffer(buffer, CL_TRUE, 0, sizeof(cl_uint) * LENGTH, &intVector[0]);
```

Display content of the vector

```
    // output contents
    std::cout << "\nContents of the buffer after kernel execution: " << std::endl;
    for (int i = 0; i < LENGTH; i++) {
        std::cout << intVector[i] << " ";
    }

}
```

output

0

```
---------------------

Please enter a multipler between 1 and 89 (inclusive): 0
Please enter a valid integer between 1 and 89.
Exiting the program...

press a key to quit...
```

1

```
Please enter a multipler between 1 and 89 (inclusive): 1

Kernel enqueued.
--------------------

Contents of the buffer after kernel execution:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 11
3 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141
142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 17
0 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 22
7 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 28
4 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312
313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 34
1 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369
370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 39
8 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426
427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 45
5 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483
484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 51
2 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540
541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 56
9 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597
598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 62
6 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654
655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 68
3 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711
712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 74
0 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768
769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 79
7 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825
826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 85
4 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882
883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 91
1 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939
940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 96
8 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996
997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 10
20 1021 1022 1023 1024
D:\1_Uni\CSCI376\A1\Task3\Debug\Task3.exe (process 12664) exited with code 0.
Press any key to close this window . . .
```

2

```
Please enter a multipler between 1 and 89 (inclusive): 2

Kernel enqueued.
--------------------

Contents of the buffer after kernel execution:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83
 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139 141 143 145 147
 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199 201 203 205 207
 209 211 213 215 217 219 221 223 225 227 229 231 233 235 237 239 241 243 245 247 249 251 253 255 257 259 261 263 265 267
 269 271 273 275 277 279 281 283 285 287 289 291 293 295 297 299 301 303 305 307 309 311 313 315 317 319 321 323 325 327
 329 331 333 335 337 339 341 343 345 347 349 351 353 355 357 359 361 363 365 367 369 371 373 375 377 379 381 383 385 387
 389 391 393 395 397 399 401 403 405 407 409 411 413 415 417 419 421 423 425 427 429 431 433 435 437 439 441 443 445 447
 449 451 453 455 457 459 461 463 465 467 469 471 473 475 477 479 481 483 485 487 489 491 493 495 497 499 501 503 505 507
 509 511 513 515 517 519 521 523 525 527 529 531 533 535 537 539 541 543 545 547 549 551 553 555 557 559 561 563 565 567
 569 571 573 575 577 579 581 583 585 587 589 591 593 595 597 599 601 603 605 607 609 611 613 615 617 619 621 623 625 627
 629 631 633 635 637 639 641 643 645 647 649 651 653 655 657 659 661 663 665 667 669 671 673 675 677 679 681 683 685 687
 689 691 693 695 697 699 701 703 705 707 709 711 713 715 717 719 721 723 725 727 729 731 733 735 737 739 741 743 745 747
 749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807
 809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867
 869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927
 929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987
 989 991 993 995 997 999 1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 1021 1023 1025 1027 1029 1031 1033 1035 1037
1039 1041 1043 1045 1047 1049 1051 1053 1055 1057 1059 1061 1063 1065 1067 1069 1071 1073 1075 1077 1079 1081 1083 1085
1087 1089 1091 1093 1095 1097 1099 1101 1103 1105 1107 1109 1111 1113 1115 1117 1119 1121 1123 1125 1127 1129 1131 1133
1135 1137 1139 1141 1143 1145 1147 1149 1151 1153 1155 1157 1159 1161 1163 1165 1167 1169 1171 1173 1175 1177 1179 1181
1183 1185 1187 1189 1191 1193 1195 1197 1199 1201 1203 1205 1207 1209 1211 1213 1215 1217 1219 1221 1223 1225 1227 1229
1231 1233 1235 1237 1239 1241 1243 1245 1247 1249 1251 1253 1255 1257 1259 1261 1263 1265 1267 1269 1271 1273 1275 1277
1279 1281 1283 1285 1287 1289 1291 1293 1295 1297 1299 1301 1303 1305 1307 1309 1311 1313 1315 1317 1319 1321 1323 1325
1327 1329 1331 1333 1335 1337 1339 1341 1343 1345 1347 1349 1351 1353 1355 1357 1359 1361 1363 1365 1367 1369 1371 1373
1375 1377 1379 1381 1383 1385 1387 1389 1391 1393 1395 1397 1399 1401 1403 1405 1407 1409 1411 1413 1415 1417 1419 1421
1423 1425 1427 1429 1431 1433 1435 1437 1439 1441 1443 1445 1447 1449 1451 1453 1455 1457 1459 1461 1463 1465 1467 1469
1471 1473 1475 1477 1479 1481 1483 1485 1487 1489 1491 1493 1495 1497 1499 1501 1503 1505 1507 1509 1511 1513 1515 1517
1519 1521 1523 1525 1527 1529 1531 1533 1535 1537 1539 1541 1543 1545 1547 1549 1551 1553 1555 1557 1559 1561 1563 1565
1567 1569 1571 1573 1575 1577 1579 1581 1583 1585 1587 1589 1591 1593 1595 1597 1599 1601 1603 1605 1607 1609 1611 1613
1615 1617 1619 1621 1623 1625 1627 1629 1631 1633 1635 1637 1639 1641 1643 1645 1647 1649 1651 1653 1655 1657 1659 1661
1663 1665 1667 1669 1671 1673 1675 1677 1679 1681 1683 1685 1687 1689 1691 1693 1695 1697 1699 1701 1703 1705 1707 1709
1711 1713 1715 1717 1719 1721 1723 1725 1727 1729 1731 1733 1735 1737 1739 1741 1743 1745 1747 1749 1751 1753 1755 1757
1759 1761 1763 1765 1767 1769 1771 1773 1775 1777 1779 1781 1783 1785 1787 1789 1791 1793 1795 1797 1799 1801 1803 1805
1807 1809 1811 1813 1815 1817 1819 1821 1823 1825 1827 1829 1831 1833 1835 1837 1839 1841 1843 1845 1847 1849 1851 1853
1855 1857 1859 1861 1863 1865 1867 1869 1871 1873 1875 1877 1879 1881 1883 1885 1887 1889 1891 1893 1895 1897 1899 1901
1903 1905 1907 1909 1911 1913 1915 1917 1919 1921 1923 1925 1927 1929 1931 1933 1935 1937 1939 1941 1943 1945 1947 1949
1951 1953 1955 1957 1959 1961 1963 1965 1967 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997
1999 2001 2003 2005 2007 2009 2011 2013 2015 2017 2019 2021 2023 2025 2027 2029 2031 2033 2035 2037 2039 2041 2043 2045
2047
D:\1_Uni\CSCI376\A1\Task3\Debug\Task3.exe (process 13092) exited with code 0.
Press any key to close this window . . .
```

3

```
Please enter a multipler between 1 and 89 (inclusive): 3

Kernel enqueued.
--------------------

Contents of the buffer after kernel execution:
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115
118 121 124 127 130 133 136 139 142 145 148 151 154 157 160 163 166 169 172 175 178 181 184 187 190 193 196 199 202 205
208 211 214 217 220 223 226 229 232 235 238 241 244 247 250 253 256 259 262 265 268 271 274 277 280 283 286 289 292 295
298 301 304 307 310 313 316 319 322 325 328 331 334 337 340 343 346 349 352 355 358 361 364 367 370 373 376 379 382 385
388 391 394 397 400 403 406 409 412 415 418 421 424 427 430 433 436 439 442 445 448 451 454 457 460 463 466 469 472 475
478 481 484 487 490 493 496 499 502 505 508 511 514 517 520 523 526 529 532 535 538 541 544 547 550 553 556 559 562 565
568 571 574 577 580 583 586 589 592 595 598 601 604 607 610 613 616 619 622 625 628 631 634 637 640 643 646 649 652 655
658 661 664 667 670 673 676 679 682 685 688 691 694 697 700 703 706 709 712 715 718 721 724 727 730 733 736 739 742 745
748 751 754 757 760 763 766 769 772 775 778 781 784 787 790 793 796 799 802 805 808 811 814 817 820 823 826 829 832 835
838 841 844 847 850 853 856 859 862 865 868 871 874 877 880 883 886 889 892 895 898 901 904 907 910 913 916 919 922 925
928 931 934 937 940 943 946 949 952 955 958 961 964 967 970 973 976 979 982 985 988 991 994 997 1000 1003 1006 1009 1012
1015 1018 1021 1024 1027 1030 1033 1036 1039 1042 1045 1048 1051 1054 1057 1060 1063 1066 1069 1072 1075 1078 1081 1084
1087 1090 1093 1096 1099 1102 1105 1108 1111 1114 1117 1120 1123 1126 1129 1132 1135 1138 1141 1144 1147 1150 1153 1156
1159 1162 1165 1168 1171 1174 1177 1180 1183 1186 1189 1192 1195 1198 1201 1204 1207 1210 1213 1216 1219 1222 1225 1228
1231 1234 1237 1240 1243 1246 1249 1252 1255 1258 1261 1264 1267 1270 1273 1276 1279 1282 1285 1288 1291 1294 1297 1300
1303 1306 1309 1312 1315 1318 1321 1324 1327 1330 1333 1336 1339 1342 1345 1348 1351 1354 1357 1360 1363 1366 1369 1372
1375 1378 1381 1384 1387 1390 1393 1396 1399 1402 1405 1408 1411 1414 1417 1420 1423 1426 1429 1432 1435 1438 1441 1444
1447 1450 1453 1456 1459 1462 1465 1468 1471 1474 1477 1480 1483 1486 1489 1492 1495 1498 1501 1504 1507 1510 1513 1516
1519 1522 1525 1528 1531 1534 1537 1540 1543 1546 1549 1552 1555 1558 1561 1564 1567 1570 1573 1576 1579 1582 1585 1588
1591 1594 1597 1600 1603 1606 1609 1612 1615 1618 1621 1624 1627 1630 1633 1636 1639 1642 1645 1648 1651 1654 1657 1660
1663 1666 1669 1672 1675 1678 1681 1684 1687 1690 1693 1696 1699 1702 1705 1708 1711 1714 1717 1720 1723 1726 1729 1732
1735 1738 1741 1744 1747 1750 1753 1756 1759 1762 1765 1768 1771 1774 1777 1780 1783 1786 1789 1792 1795 1798 1801 1804
1807 1810 1813 1816 1819 1822 1825 1828 1831 1834 1837 1840 1843 1846 1849 1852 1855 1858 1861 1864 1867 1870 1873 1876
1879 1882 1885 1888 1891 1894 1897 1900 1903 1906 1909 1912 1915 1918 1921 1924 1927 1930 1933 1936 1939 1942 1945 1948
1951 1954 1957 1960 1963 1966 1969 1972 1975 1978 1981 1984 1987 1990 1993 1996 1999 2002 2005 2008 2011 2014 2017 2020
2023 2026 2029 2032 2035 2038 2041 2044 2047 2050 2053 2056 2059 2062 2065 2068 2071 2074 2077 2080 2083 2086 2089 2092
2095 2098 2101 2104 2107 2110 2113 2116 2119 2122 2125 2128 2131 2134 2137 2140 2143 2146 2149 2152 2155 2158 2161 2164
2167 2170 2173 2176 2179 2182 2185 2188 2191 2194 2197 2200 2203 2206 2209 2212 2215 2218 2221 2224 2227 2230 2233 2236
2239 2242 2245 2248 2251 2254 2257 2260 2263 2266 2269 2272 2275 2278 2281 2284 2287 2290 2293 2296 2299 2302 2305 2308
2311 2314 2317 2320 2323 2326 2329 2332 2335 2338 2341 2344 2347 2350 2353 2356 2359 2362 2365 2368 2371 2374 2377 2380
2383 2386 2389 2392 2395 2398 2401 2404 2407 2410 2413 2416 2419 2422 2425 2428 2431 2434 2437 2440 2443 2446 2449 2452
2455 2458 2461 2464 2467 2470 2473 2476 2479 2482 2485 2488 2491 2494 2497 2500 2503 2506 2509 2512 2515 2518 2521 2524
2527 2530 2533 2536 2539 2542 2545 2548 2551 2554 2557 2560 2563 2566 2569 2572 2575 2578 2581 2584 2587 2590 2593 2596
2599 2602 2605 2608 2611 2614 2617 2620 2623 2626 2629 2632 2635 2638 2641 2644 2647 2650 2653 2656 2659 2662 2665 2668
2671 2674 2677 2680 2683 2686 2689 2692 2695 2698 2701 2704 2707 2710 2713 2716 2719 2722 2725 2728 2731 2734 2737 2740
2743 2746 2749 2752 2755 2758 2761 2764 2767 2770 2773 2776 2779 2782 2785 2788 2791 2794 2797 2800 2803 2806 2809 2812
2815 2818 2821 2824 2827 2830 2833 2836 2839 2842 2845 2848 2851 2854 2857 2860 2863 2866 2869 2872 2875 2878 2881 2884
2887 2890 2893 2896 2899 2902 2905 2908 2911 2914 2917 2920 2923 2926 2929 2932 2935 2938 2941 2944 2947 2950 2953 2956
2959 2962 2965 2968 2971 2974 2977 2980 2983 2986 2989 2992 2995 2998 3001 3004 3007 3010 3013 3016 3019 3022 3025 3028
3031 3034 3037 3040 3043 3046 3049 3052 3055 3058 3061 3064 3067 3070
D:\1_Uni\CSCI376\A1\Task3\Debug\Task3.exe (process 10036) exited with code 0.
Press any key to close this window . . .
```

89

```
Please enter a multipler between 1 and 89 (inclusive): 89

Kernel enqueued.
--------------------

Contents of the buffer after kernel execution:
1 90 179 268 357 446 535 624 713 802 891 980 1069 1158 1247 1336 1425 1514 1603 1692 1781 1870 1959 2048 2137 2226 2315
2404 2493 2582 2671 2760 2849 2938 3027 3116 3205 3294 3383 3472 3561 3650 3739 3828 3917 4006 4095 4184 4273 4362 4451
4540 4629 4718 4807 4896 4985 5074 5163 5252 5341 5430 5519 5608 5697 5786 5875 5964 6053 6142 6231 6320 6409 6498 6587
6676 6765 6854 6943 7032 7121 7210 7299 7388 7477 7566 7655 7744 7833 7922 8011 8100 8189 8278 8367 8456 8545 8634 8723
8812 8901 8990 9079 9168 9257 9346 9435 9524 9613 9702 9791 9880 9969 10058 10147 10236 10325 10414 10503 10592 10681 10
770 10859 10948 11037 11126 11215 11304 11393 11482 11571 11660 11749 11838 11927 12016 12105 12194 12283 12372 12461 12
550 12639 12728 12817 12906 12995 13084 13173 13262 13351 13440 13529 13618 13707 13796 13885 13974 14063 14152 14241 14
330 14419 14508 14597 14686 14775 14864 14953 15042 15131 15220 15309 15398 15487 15576 15665 15754 15843 15932 16021 16
110 16199 16288 16377 16466 16555 16644 16733 16822 16911 17000 17089 17178 17267 17356 17445 17534 17623 17712 17801 17
890 17979 18068 18157 18246 18335 18424 18513 18602 18691 18780 18869 18958 19047 19136 19225 19314 19403 19492 19581 19
670 19759 19848 19937 20026 20115 20204 20293 20382 20471 20560 20649 20738 20827 20916 21005 21094 21183 21272 21361 21
450 21539 21628 21717 21806 21895 21984 22073 22162 22251 22340 22429 22518 22607 22696 22785 22874 22963 23052 23141 23
230 23319 23408 23497 23586 23675 23764 23853 23942 24031 24120 24209 24298 24387 24476 24565 24654 24743 24832 24921 25
010 25099 25188 25277 25366 25455 25544 25633 25722 25811 25900 25989 26078 26167 26256 26345 26434 26523 26612 26701 26
790 26879 26968 27057 27146 27235 27324 27413 27502 27591 27680 27769 27858 27947 28036 28125 28214 28303 28392 28481 28
570 28659 28748 28837 28926 29015 29104 29193 29282 29371 29460 29549 29638 29727 29816 29905 29994 30083 30172 30261 30
350 30439 30528 30617 30706 30795 30884 30973 31062 31151 31240 31329 31418 31507 31596 31685 31774 31863 31952 32041 32
130 32219 32308 32397 32486 32575 32664 32753 32842 32931 33020 33109 33198 33287 33376 33465 33554 33643 33732 33821 33
910 33999 34088 34177 34266 34355 34444 34533 34622 34711 34800 34889 34978 35067 35156 35245 35334 35423 35512 35601 35
690 35779 35868 35957 36046 36135 36224 36313 36402 36491 36580 36669 36758 36847 36936 37025 37114 37203 37292 37381 37
470 37559 37648 37737 37826 37915 38004 38093 38182 38271 38360 38449 38538 38627 38716 38805 38894 38983 39072 39161 39
250 39339 39428 39517 39606 39695 39784 39873 39962 40051 40140 40229 40318 40407 40496 40585 40674 40763 40852 40941 41
030 41119 41208 41297 41386 41475 41564 41653 41742 41831 41920 42009 42098 42187 42276 42365 42454 42543 42632 42721 42
810 42899 42988 43077 43166 43255 43344 43433 43522 43611 43700 43789 43878 43967 44056 44145 44234 44323 44412 44501 44
590 44679 44768 44857 44946 45035 45124 45213 45302 45391 45480 45569 45658 45747 45836 45925 46014 46103 46192 46281 46
370 46459 46548 46637 46726 46815 46904 46993 47082 47171 47260 47349 47438 47527 47616 47705 47794 47883 47972 48061 48
150 48239 48328 48417 48506 48595 48684 48773 48862 48951 49040 49129 49218 49307 49396 49485 49574 49663 49752 49841 49
930 50019 50108 50197 50286 50375 50464 50553 50642 50731 50820 50909 50998 51087 51176 51265 51354 51443 51532 51621 51
710 51799 51888 51977 52066 52155 52244 52333 52422 52511 52600 52689 52778 52867 52956 53045 53134 53223 53312 53401 53
490 53579 53668 53757 53846 53935 54024 54113 54202 54291 54380 54469 54558 54647 54736 54825 54914 55003 55092 55181 55
270 55359 55448 55537 55626 55715 55804 55893 55982 56071 56160 56249 56338 56427 56516 56605 56694 56783 56872 56961 57
050 57139 57228 57317 57406 57495 57584 57673 57762 57851 57940 58029 58118 58207 58296 58385 58474 58563 58652 58741 58
830 58919 59008 59097 59186 59275 59364 59453 59542 59631 59720 59809 59898 59987 60076 60165 60254 60343 60432 60521 60
610 60699 60788 60877 60966 61055 61144 61233 61322 61411 61500 61589 61678 61767 61856 61945 62034 62123 62212 62301 62
390 62479 62568 62657 62746 62835 62924 63013 63102 63191 63280 63369 63458 63547 63636 63725 63814 63903 63992 64081 64
170 64259 64348 64437 64526 64615 64704 64793 64882 64971 65060 65149 65238 65327 65416 65505 65594 65683 65772 65861 65
950 66039 66128 66217 66306 66395 66484 66573 66662 66751 66840 66929 67018 67107 67196 67285 67374 67463 67552 67641 67
730 67819 67908 67997 68086 68175 68264 68353 68442 68531 68620 68709 68798 68887 68976 69065 69154 69243 69332 69421 69
510 69599 69688 69777 69866 69955 70044 70133 70222 70311 70400 70489 70578 70667 70756 70845 70934 71023 71112 71201 71
290 71379 71468 71557 71646 71735 71824 71913 72002 72091 72180 72269 72358 72447 72536 72625 72714 72803 72892 72981 73
070 73159 73248 73337 73426 73515 73604 73693 73782 73871 73960 74049 74138 74227 74316 74405 74494 74583 74672 74761 74
850 74939 75028 75117 75206 75295 75384 75473 75562 75651 75740 75829 75918 76007 76096 76185 76274 76363 76452 76541 76
630 76719 76808 76897 76986 77075 77164 77253 77342 77431 77520 77609 77698 77787 77876 77965 78054 78143 78232 78321 78
410 78499 78588 78677 78766 78855 78944 79033 79122 79211 79300 79389 79478 79567 79656 79745 79834 79923 80012 80101 80
190 80279 80368 80457 80546 80635 80724 80813 80902 80991 81080 81169 81258 81347 81436 81525 81614 81703 81792 81881 81
970 82059 82148 82237 82326 82415 82504 82593 82682 82771 82860 82949 83038 83127 83216 83305 83394 83483 83572 83661 83
750 83839 83928 84017 84106 84195 84284 84373 84462 84551 84640 84729 84818 84907 84996 85085 85174 85263 85352 85441 85
530 85619 85708 85797 85886 85975 86064 86153 86242 86331 86420 86509 86598 86687 86776 86865 86954 87043 87132 87221 87
310 87399 87488 87577 87666 87755 87844 87933 88022 88111 88200 88289 88378 88467 88556 88645 88734 88823 88912 89001 89
090 89179 89268 89357 89446 89535 89624 89713 89802 89891 89980 90069 90158 90247 90336 90425 90514 90603 90692 90781 90
870 90959 91048
D:\1_Uni\CSCI376\A1\Task3\Debug\Task3.exe (process 14164) exited with code 0.
Press any key to close this window
```