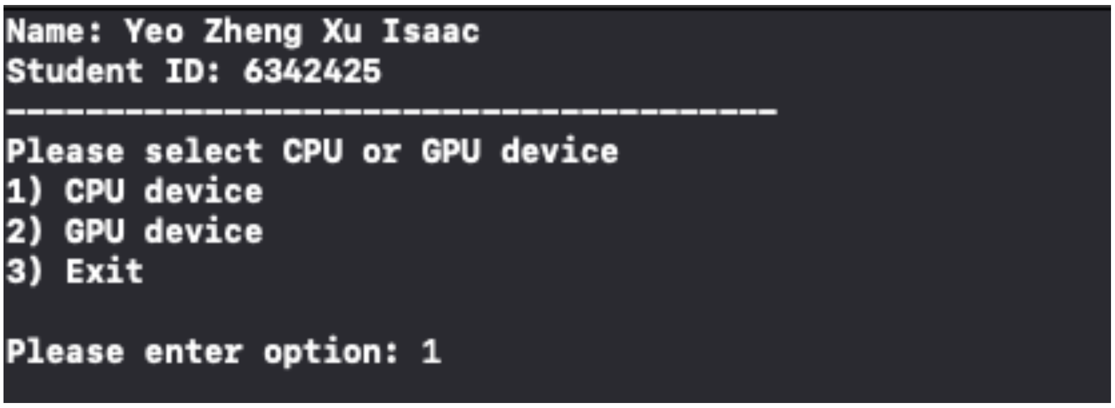
Assignment 1 report

Name: Yeo Zheng Xu Isaac

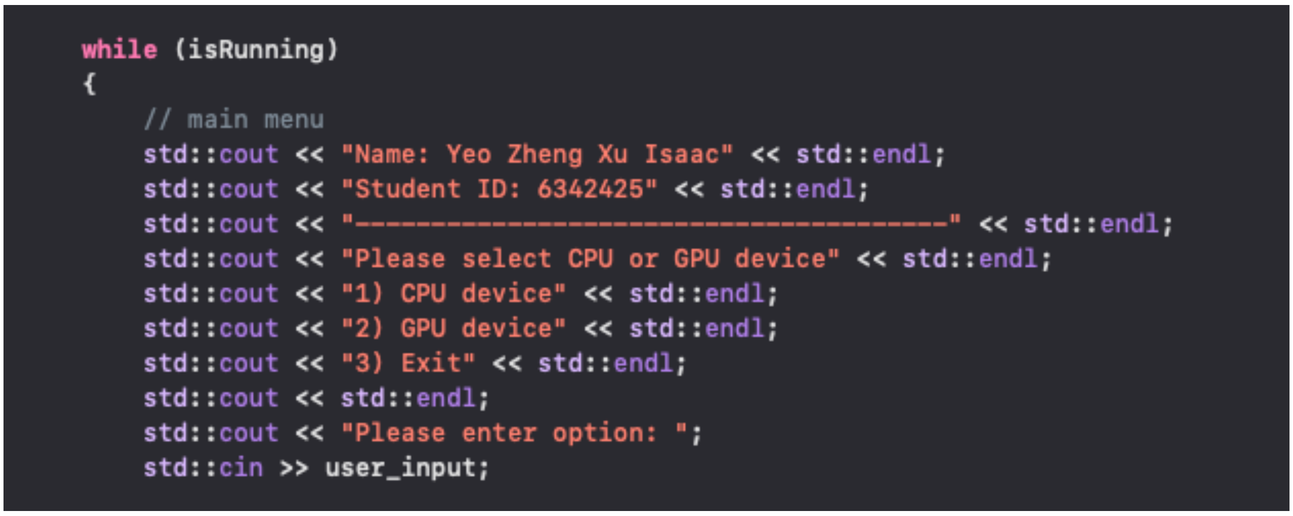
TASK 1 - Device Information

The following screenshots displays the code explanations & task program for the following tasks requirements:

- 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).



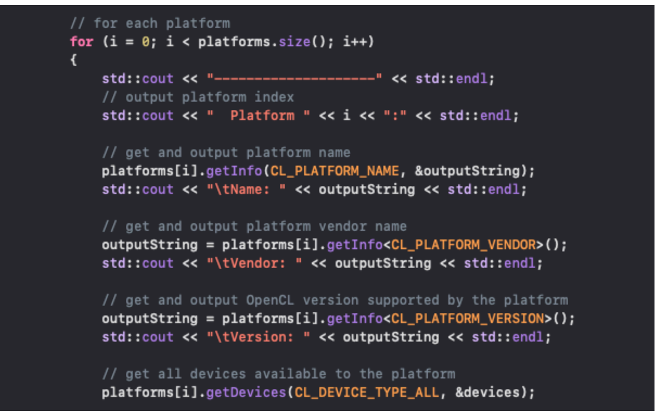
```
Name: Yeo Zheng Xu Isaac
Student ID: 6342425
---------------------------------------
Please select CPU or GPU device
1) CPU device
2) GPU device
3) Exit

Please enter option: 1
```



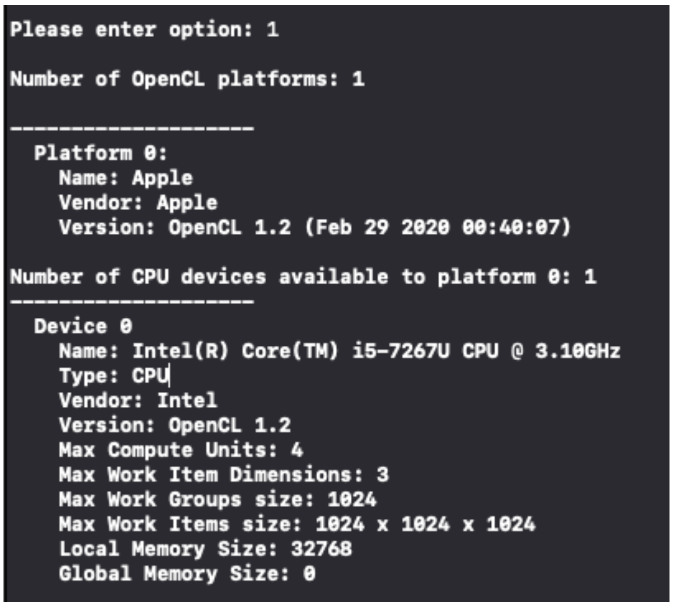
```
    while (isRunning)
    {
        // main menu
        std::cout << "Name: Yeo Zheng Xu Isaac" << std::endl;
        std::cout << "Student ID: 6342425" << std::endl;
        std::cout << "---------------------------------------" << std::endl;
        std::cout << "Please select CPU or GPU device" << std::endl;
        std::cout << "1) CPU device" << std::endl;
        std::cout << "2) GPU device" << std::endl;
        std::cout << "3) Exit" << std::endl;
        std::cout << std::endl;
        std::cout << "Please enter option: ";
        std::cin >> user_input;
```

- Based on the user’s choice, display the following information for each CPU/GPU device that is available on the system:

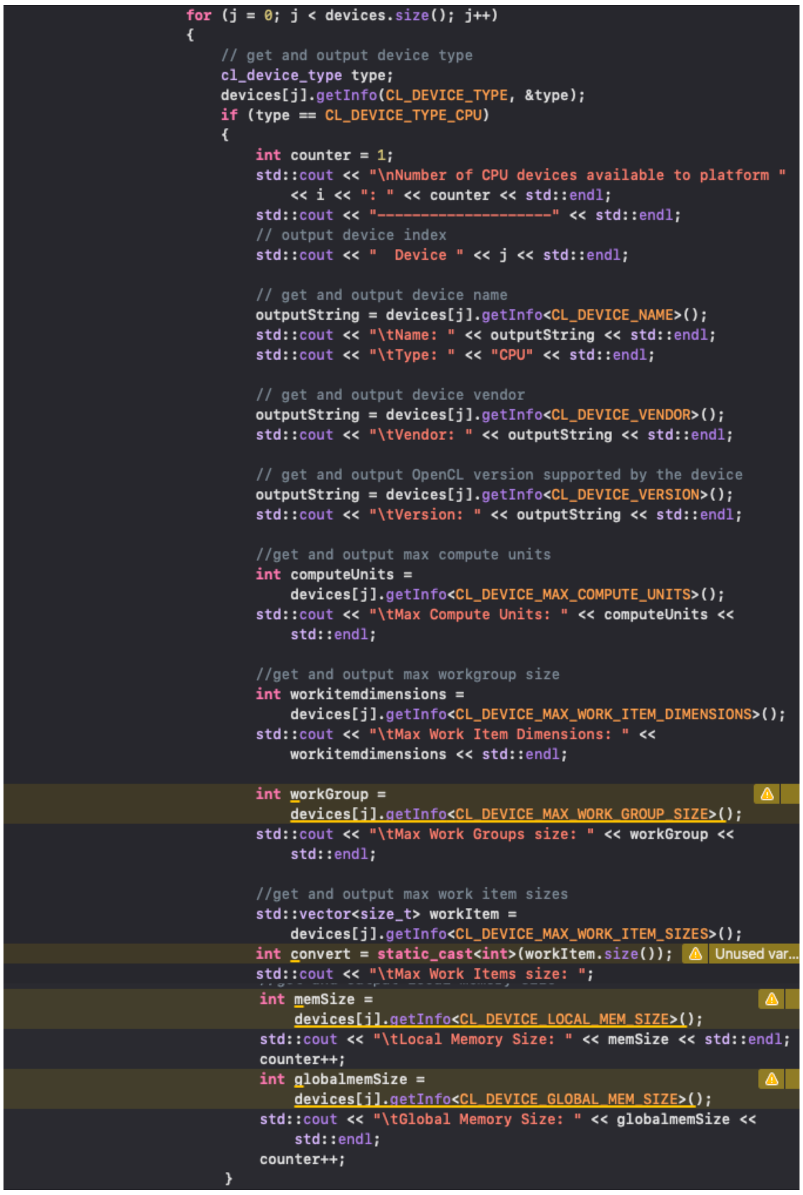
- o Name of the platform that supports that device
- o Device type – CPU or GPU
  (CL_DEVICE_TYPE – either CL_DEVICE_TYPE _CPU or CL_DEVICE_TYPE_GPU)
- o Device name (CL_DEVICE_NAME)
- o Number of compute units (CL_DEVICE_MAX_COMPUTE_UNITS)
- o Maximum work group size (CL_DEVICE_MAX_WORK_GROUP_SIZE)
- o Maximum number of work item dimensions (CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS)
- o Maximum work item sizes (CL_DEVICE_MAX_WORK_ITEM_SIZES)
- o Global memory size (CL_DEVICE_GLOBAL_MEM_SIZE)
- o Local memory size
  (CL_DEVICE_LOCAL_MEM_SIZE)



```cpp
    // for each platform
    for (i = 0; i < platforms.size(); i++)
    {
        std::cout << "--------------------" << std::endl;
        // output platform index
        std::cout << "  Platform " << i << ":" << std::endl;

        // get and output platform name
        platforms[i].getInfo(CL_PLATFORM_NAME, &outputString);
        std::cout << "\tName: " << outputString << std::endl;

        // get and output platform vendor name
        outputString = platforms[i].getInfo<CL_PLATFORM_VENDOR>();
        std::cout << "\tVendor: " << outputString << std::endl;

        // get and output OpenCL version supported by the platform
        outputString = platforms[i].getInfo<CL_PLATFORM_VERSION>();
        std::cout << "\tVersion: " << outputString << std::endl;

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

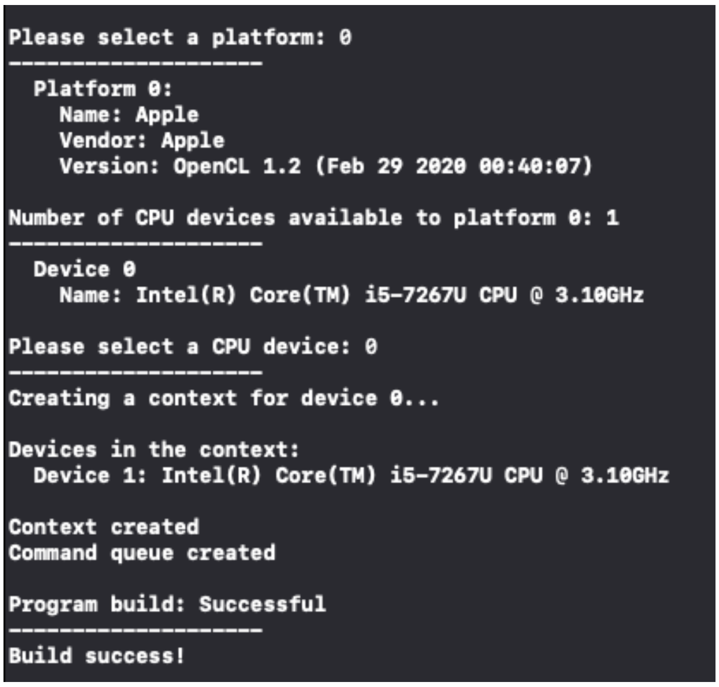


```
Please enter option: 1

Number of OpenCL platforms: 1

--------------------
  Platform 0:
    Name: Apple
    Vendor: Apple
    Version: OpenCL 1.2 (Feb 29 2020 00:40:07)

Number of CPU devices available to platform 0: 1
--------------------
  Device 0
    Name: Intel(R) Core(TM) i5-7267U CPU @ 3.10GHz
    Type: CPU
    Vendor: Intel
    Version: OpenCL 1.2
    Max Compute Units: 4
    Max Work Item Dimensions: 3
    Max Work Groups size: 1024
    Max Work Items size: 1024 x 1024 x 1024
    Local Memory Size: 32768
    Global Memory Size: 0
```

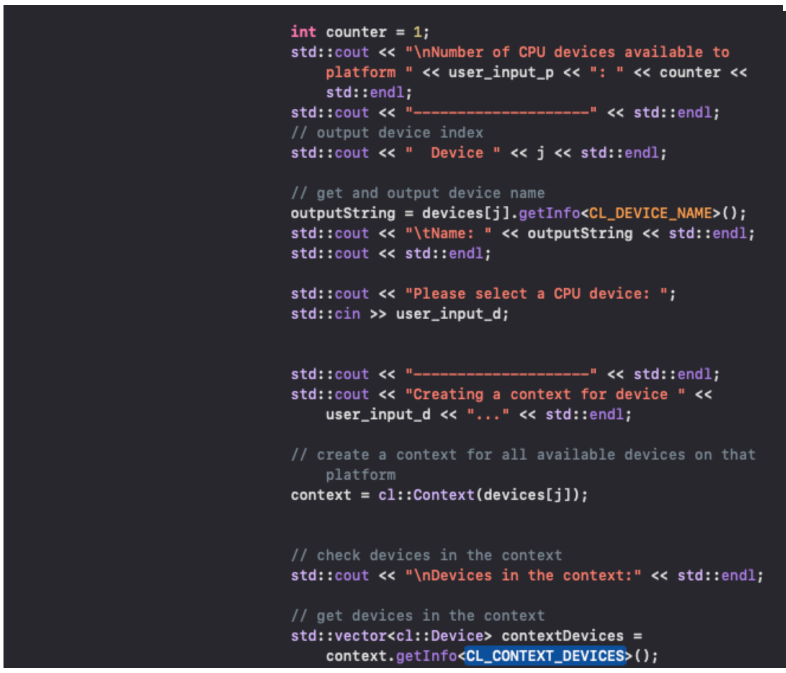


```cpp
        for (j = 0; j < devices.size(); j++)
        {
            // get and output device type
            cl_device_type type;
            devices[j].getInfo(CL_DEVICE_TYPE, &type);
            if (type == CL_DEVICE_TYPE_CPU)
            {
                int counter = 1;
                std::cout << "\nNumber of CPU devices available to platform "
                    << i << ": " << counter << std::endl;
                std::cout << "--------------------" << std::endl;
                // output device index
                std::cout << "  Device " << j << std::endl;

                // get and output device name
                outputString = devices[j].getInfo<CL_DEVICE_NAME>();
                std::cout << "\tName: " << outputString << std::endl;
                std::cout << "\tType: " << "CPU" << std::endl;

                // get and output device vendor
                outputString = devices[j].getInfo<CL_DEVICE_VENDOR>();
                std::cout << "\tVendor: " << outputString << std::endl;

                // get and output OpenCL version supported by the device
                outputString = devices[j].getInfo<CL_DEVICE_VERSION>();
                std::cout << "\tVersion: " << outputString << std::endl;

                //get and output max compute units
                int computeUnits =
                    devices[j].getInfo<CL_DEVICE_MAX_COMPUTE_UNITS>();
                std::cout << "\tMax Compute Units: " << computeUnits <<
                    std::endl;

                //get and output max workgroup size
                int workitemdimensions =
                    devices[j].getInfo<CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS>();
                std::cout << "\tMax Work Item Dimensions: " <<
                    workitemdimensions << std::endl;

                int workGroup =
                    devices[j].getInfo<CL_DEVICE_MAX_WORK_GROUP_SIZE>();
                std::cout << "\tMax Work Groups size: " << workGroup <<
                    std::endl;

                //get and output max work item sizes
                std::vector<size_t> workItem =
                    devices[j].getInfo<CL_DEVICE_MAX_WORK_ITEM_SIZES>();
                int convert = static_cast<int>(workItem.size());  ⚠ Unused var...
                std::cout << "\tMax Work Items size: ";
                int memSize =
                    devices[j].getInfo<CL_DEVICE_LOCAL_MEM_SIZE>();
                std::cout << "\tLocal Memory Size: " << memSize << std::endl;
                counter++;
                int globalmemSize =
                    devices[j].getInfo<CL_DEVICE_GLOBAL_MEM_SIZE>();
                std::cout << "\tGlobal Memory Size: " << globalmemSize <<
                    std::endl;
                counter++;
            }
```

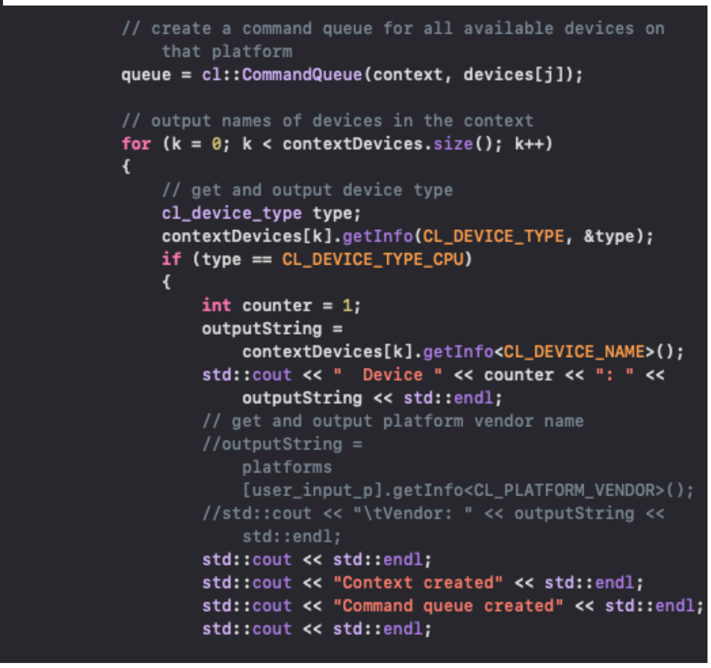
- Based on the devices available, allow the user to select one device. Create a context using that device, and a command queue.



```
Please select a platform: 0
--------------------
  Platform 0:
    Name: Apple
    Vendor: Apple
    Version: OpenCL 1.2 (Feb 29 2020 00:40:07)

Number of CPU devices available to platform 0: 1
--------------------
  Device 0
    Name: Intel(R) Core(TM) i5-7267U CPU @ 3.10GHz

Please select a CPU device: 0
--------------------
Creating a context for device 0...

Devices in the context:
  Device 1: Intel(R) Core(TM) i5-7267U CPU @ 3.10GHz

Context created
Command queue created

Program build: Successful
--------------------
Build success!
```

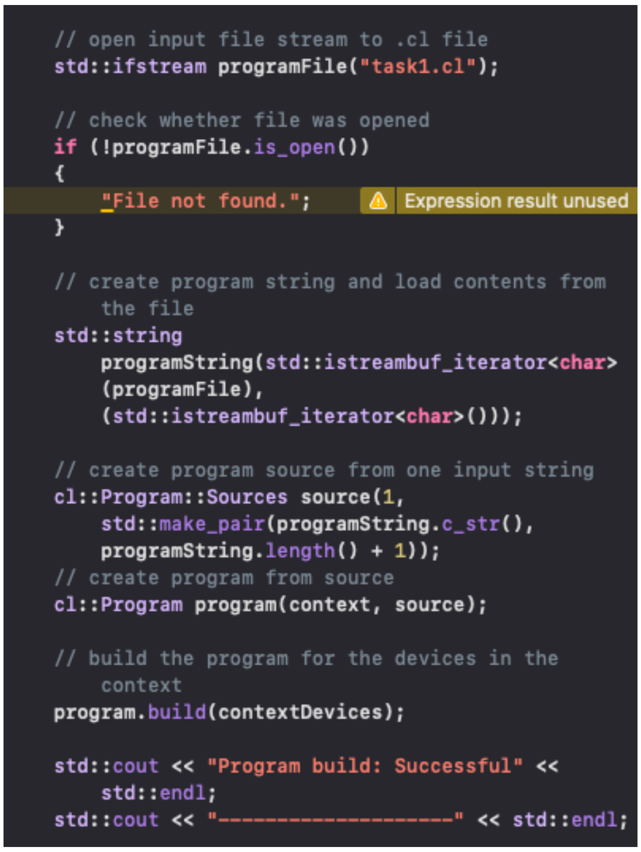


```cpp
int counter = 1;
std::cout << "\nNumber of CPU devices available to
    platform " << user_input_p << ": " << counter <<
    std::endl;
std::cout << "--------------------" << std::endl;
// output device index
std::cout << "  Device " << j << std::endl;

// get and output device name
outputString = devices[j].getInfo<CL_DEVICE_NAME>();
std::cout << "\tName: " << outputString << std::endl;
std::cout << std::endl;

std::cout << "Please select a CPU device: ";
std::cin >> user_input_d;


std::cout << "--------------------" << std::endl;
std::cout << "Creating a context for device " <<
    user_input_d << "..." << std::endl;

// create a context for all available devices on that
    platform
context = cl::Context(devices[j]);


// check devices in the context
std::cout << "\nDevices in the context:" << std::endl;

// get devices in the context
std::vector<cl::Device> contextDevices =
    context.getInfo<CL_CONTEXT_DEVICES>();
```

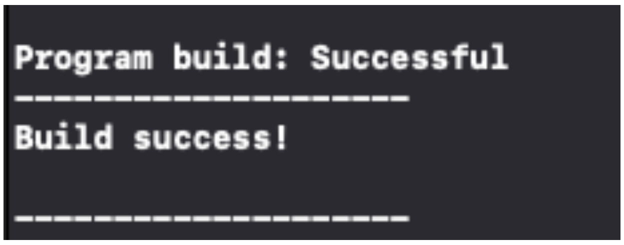


```cpp
// create a command queue for all available devices on
    that platform
queue = cl::CommandQueue(context, devices[j]);

// output names of devices in the context
for (k = 0; k < contextDevices.size(); k++)
{
    // get and output device type
    cl_device_type type;
    contextDevices[k].getInfo(CL_DEVICE_TYPE, &type);
    if (type == CL_DEVICE_TYPE_CPU)
    {
        int counter = 1;
        outputString =
            contextDevices[k].getInfo<CL_DEVICE_NAME>();
        std::cout << "  Device " << counter << ": " <<
            outputString << std::endl;
        // get and output platform vendor name
        //outputString =
            platforms
            [user_input_p].getInfo<CL_PLATFORM_VENDOR>();
        //std::cout << "\tVendor: " << outputString <<
            std::endl;
        std::cout << std::endl;
        std::cout << "Context created" << std::endl;
        std::cout << "Command queue created" << std::endl;
        std::cout << std::endl;
```

- 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 (display the build log even if the program built successfully).

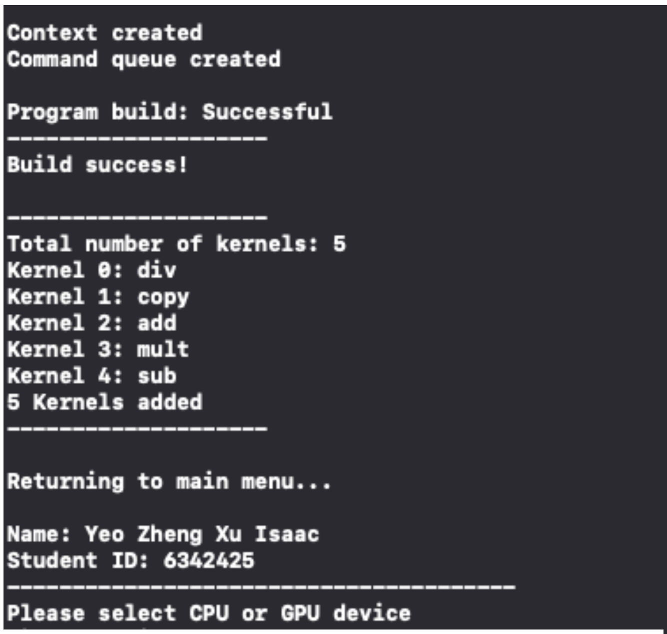


```
// open input file stream to .cl file
std::ifstream programFile("task1.cl");

// check whether file was opened
if (!programFile.is_open())
{
    "File not found.";      ⚠ Expression result unused
}

// 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
cl::Program program(context, source);

// build the program for the devices in the
    context
program.build(contextDevices);

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

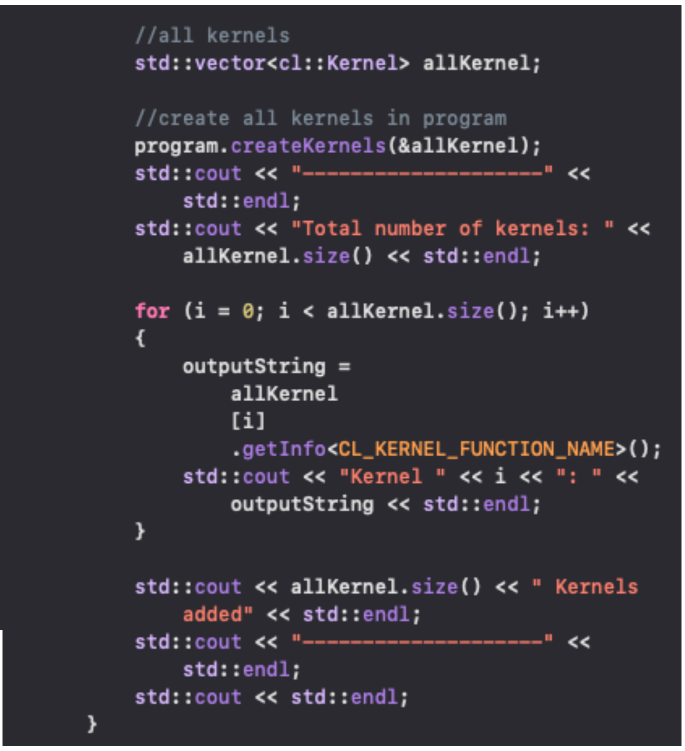


```
Program build: Successful
--------------------
Build success!

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

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



```
Context created
Command queue created

Program build: Successful
--------------------
Build success!

--------------------
Total number of kernels: 5
Kernel 0: div
Kernel 1: copy
Kernel 2: add
Kernel 3: mult
Kernel 4: sub
5 Kernels added
--------------------

Returning to main menu...

Name: Yeo Zheng Xu Isaac
Student ID: 6342425
----------------------------------------
Please select CPU or GPU device
```

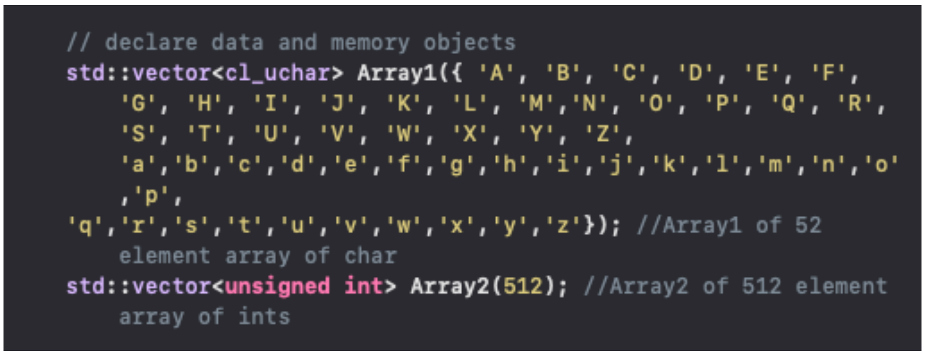


```
    //all kernels
    std::vector<cl::Kernel> allKernel;

    //create all kernels in program
    program.createKernels(&allKernel);
    std::cout << "--------------------" <<
        std::endl;
    std::cout << "Total number of kernels: " <<
        allKernel.size() << std::endl;

    for (i = 0; i < allKernel.size(); i++)
    {
        outputString =
            allKernel
            [i]
            .getInfo<CL_KERNEL_FUNCTION_NAME>();
        std::cout << "Kernel " << i << ": " <<
            outputString << std::endl;
    }

    std::cout << allKernel.size() << " Kernels
        added" << std::endl;
    std::cout << "--------------------" <<
        std::endl;
    std::cout << std::endl;
}
```

TASK 2 - Data Management

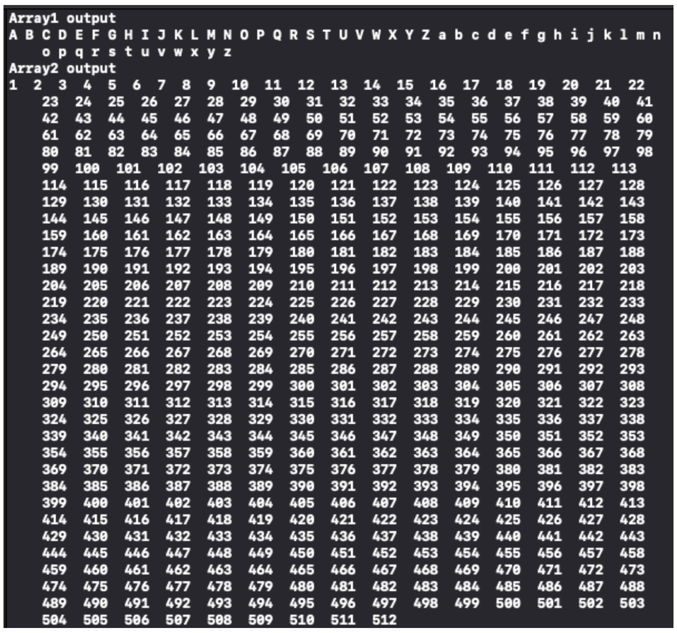
- Create a C++ vector of unsigned chars to store alphabets. Initialise its contents to: a-z and A-Z (i.e. 52 alphabets in total). Create another C++ vector to store 512 unsigned ints. Initialise its contents to: 1-512.

//Creating 2 vectors for both 52 alphabets and 512 unsigned ints respectively.



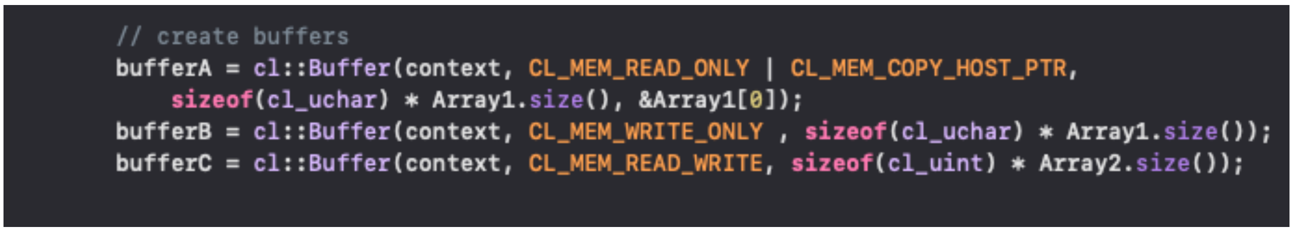
```
// declare data and memory objects
std::vector<cl_uchar> Array1({ 'A', 'B', 'C', 'D', 'E', 'F',
    'G', 'H', 'I', 'J', 'K', 'L', 'M','N', 'O', 'P', 'Q', 'R',
    'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
    'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o'
    ,'p',
'q','r','s','t','u','v','w','x','y','z'}); //Array1 of 52
    element array of char
std::vector<unsigned int> Array2(512); //Array2 of 512 element
    array of ints
```

//Initialising arrays and displaying both arrays in program.



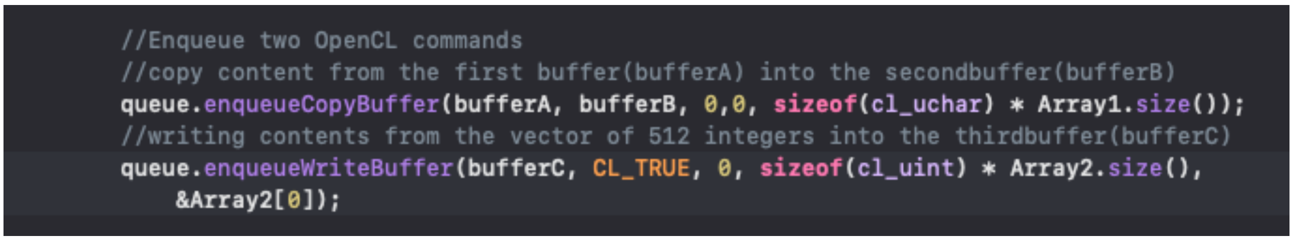
```
Array1 output
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n
    o p q r s t u v w x y z
Array2 output
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22
    23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41
    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
```

- 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.



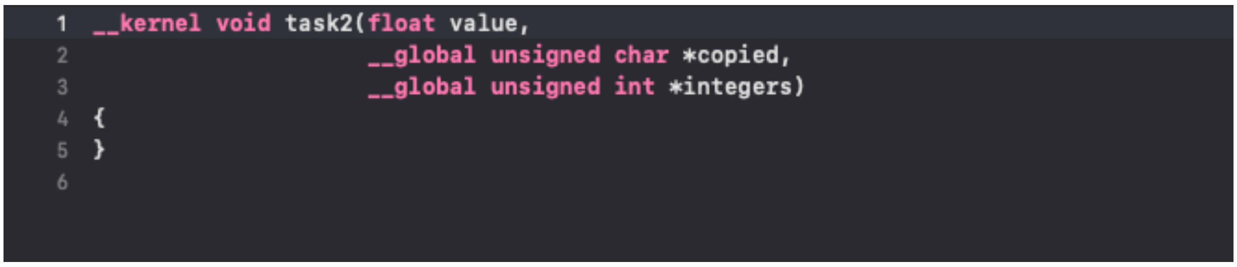
```
// create buffers
bufferA = cl::Buffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
    sizeof(cl_uchar) * Array1.size(), &Array1[0]);
bufferB = cl::Buffer(context, CL_MEM_WRITE_ONLY , sizeof(cl_uchar) * Array1.size());
bufferC = cl::Buffer(context, CL_MEM_READ_WRITE, sizeof(cl_uint) * Array2.size());
```

- 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.

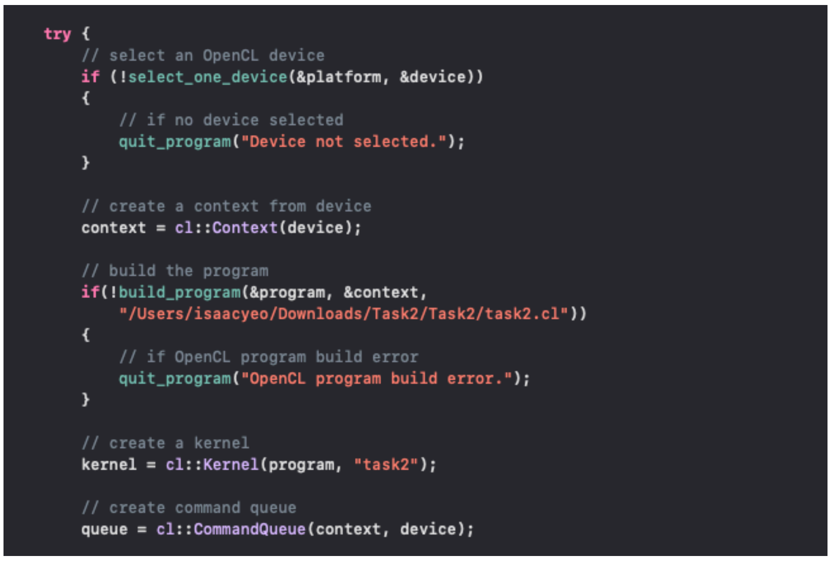


```
//Enqueue two OpenCL commands
//copy content from the first buffer(bufferA) into the secondbuffer(bufferB)
queue.enqueueCopyBuffer(bufferA, bufferB, 0,0, sizeof(cl_uchar) * Array1.size());
//writing contents from the vector of 512 integers into the thirdbuffer(bufferC)
queue.enqueueWriteBuffer(bufferC, CL_TRUE, 0, sizeof(cl_uint) * Array2.size(),
    &Array2[0]);
```

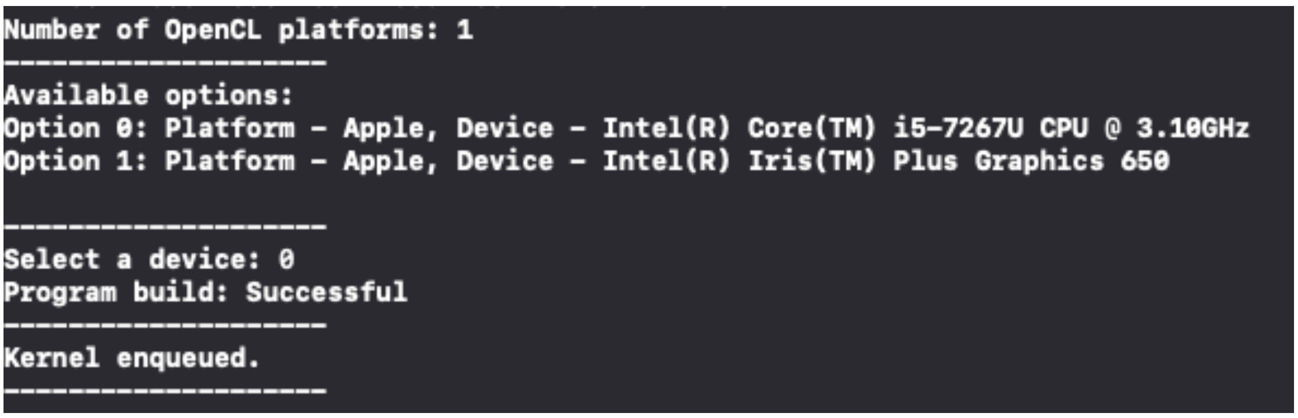
- Setup the OpenCL program to allow the user to select one device, create a context and command queue for that device. Then, build the provided "task2.cl" program and create a kernel for "task2".



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

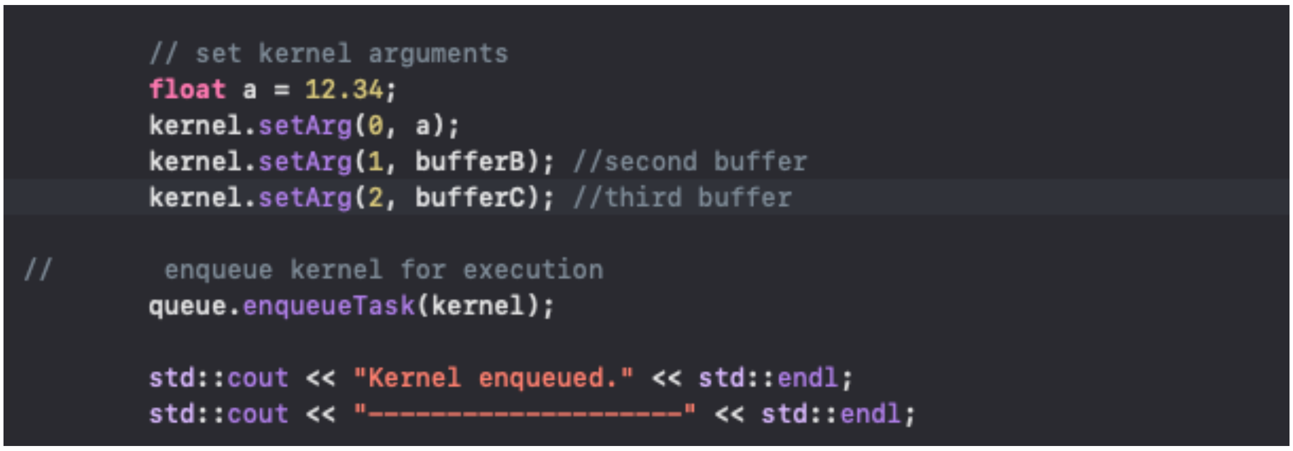


```
try {
    // select an OpenCL device
    if (!select_one_device(&platform, &device))
    {
        // if no device selected
        quit_program("Device not selected.");
    }

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

    // build the program
    if(!build_program(&program, &context,
        "/Users/isaacyeo/Downloads/Task2/Task2/task2.cl"))
    {
        // if OpenCL program build error
        quit_program("OpenCL program build error.");
    }

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

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



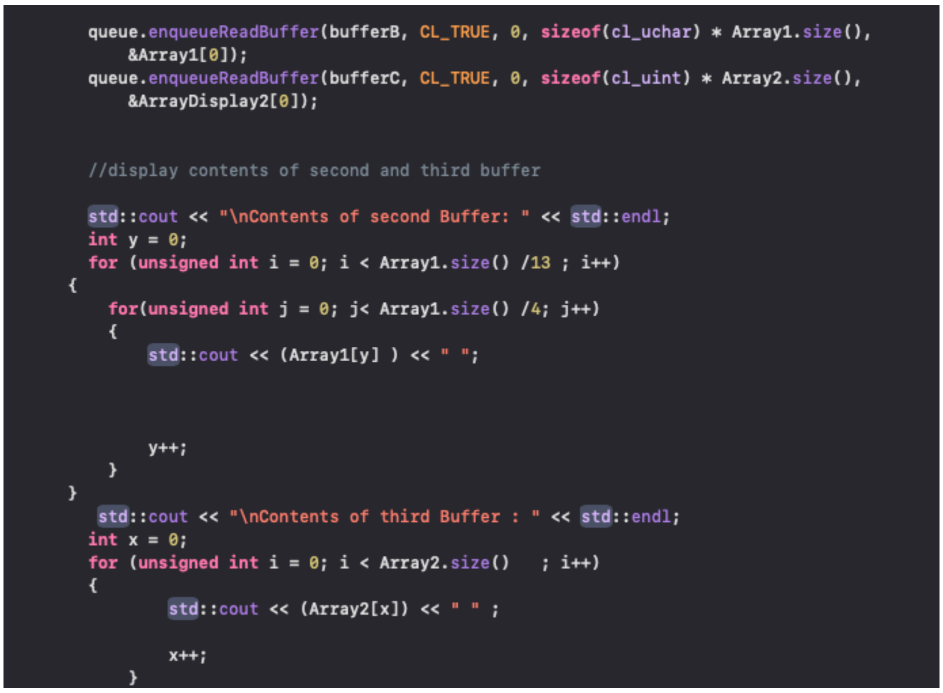
```
Number of OpenCL platforms: 1
--------------------
Available options:
Option 0: Platform - Apple, Device - Intel(R) Core(TM) i5-7267U CPU @ 3.10GHz
Option 1: Platform - Apple, Device - Intel(R) Iris(TM) Plus Graphics 650

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

- 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.



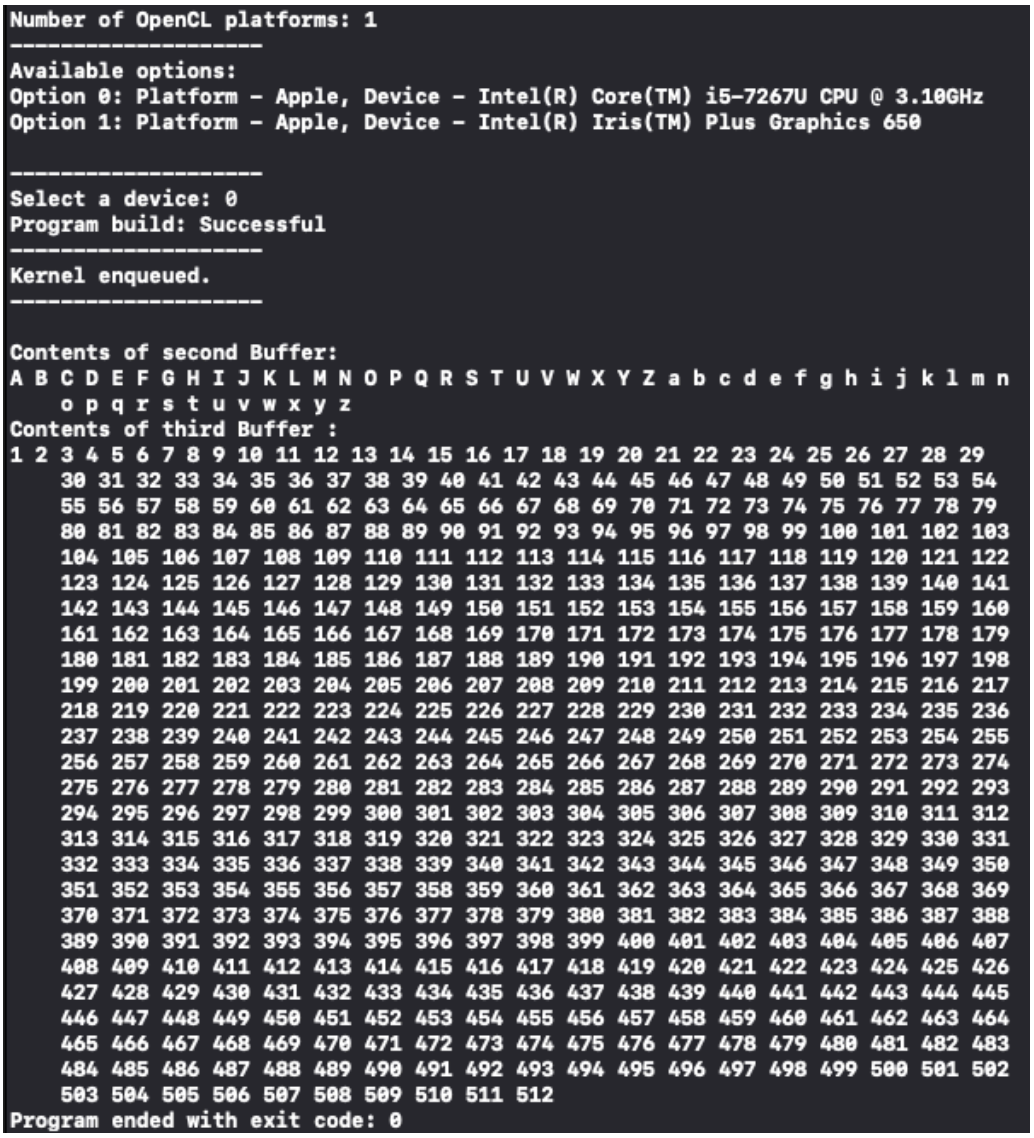
```
        // set kernel arguments
        float a = 12.34;
        kernel.setArg(0, a);
        kernel.setArg(1, bufferB); //second buffer
        kernel.setArg(2, bufferC); //third buffer

//       enqueue kernel for execution
        queue.enqueueTask(kernel);

        std::cout << "Kernel enqueued." << std::endl;
        std::cout << "--------------------" << std::endl;
```

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



```
        queue.enqueueReadBuffer(bufferB, CL_TRUE, 0, sizeof(cl_uchar) * Array1.size(),
            &Array1[0]);
        queue.enqueueReadBuffer(bufferC, CL_TRUE, 0, sizeof(cl_uint) * Array2.size(),
            &ArrayDisplay2[0]);


        //display contents of second and third buffer

        std::cout << "\nContents of second Buffer: " << std::endl;
        int y = 0;
        for (unsigned int i = 0; i < Array1.size() /13 ; i++)
      {
          for(unsigned int j = 0; j< Array1.size() /4; j++)
          {
              std::cout << (Array1[y] ) << " ";



              y++;
          }
      }
         std::cout << "\nContents of third Buffer : " << std::endl;
        int x = 0;
        for (unsigned int i = 0; i < Array2.size()   ; i++)
        {
                std::cout << (Array2[x]) << " " ;

                x++;
            }
```

Program full run



```
Number of OpenCL platforms: 1
--------------------
Available options:
Option 0: Platform - Apple, Device - Intel(R) Core(TM) i5-7267U CPU @ 3.10GHz
Option 1: Platform - Apple, Device - Intel(R) Iris(TM) Plus Graphics 650

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

Contents of second Buffer:
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 :
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
Program ended with exit code: 0
```

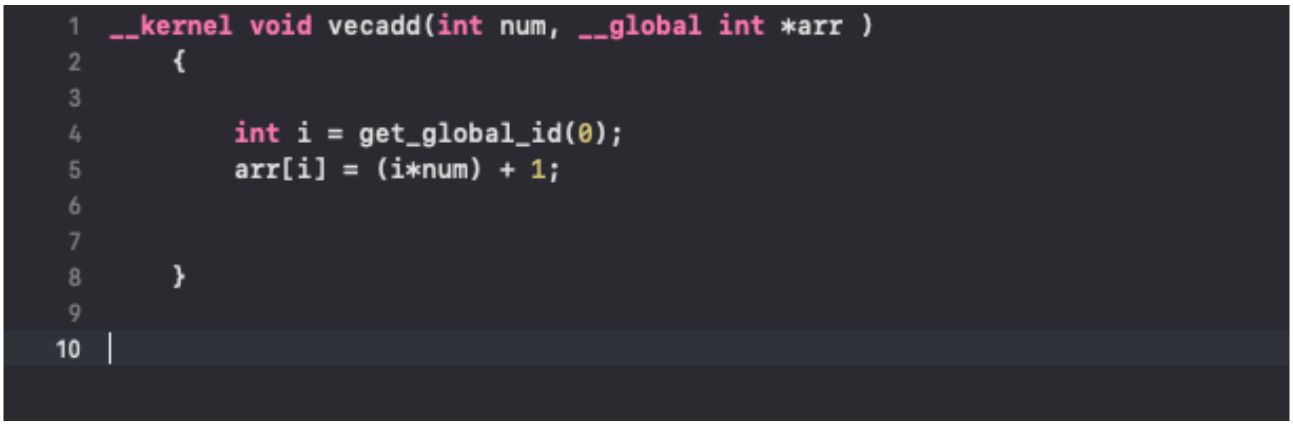
TASK 3 - Kernel Execution

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 100 (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:

- If the user enters 1, the resulting contents of the array should be: 1, 2, 3, 4, 5,... until 1024
- If the user enters 2, the resulting contents of the array should be: 1, 3, 5, 7, 9,... until 2047
- If the user enters 3, the resulting contents of the array should be: 1, 4, 7, 10, 13,... until 3070
- If the user enters 100, the resulting contents of the array should be: 1, 101, 201, 301, 401,... until 102301

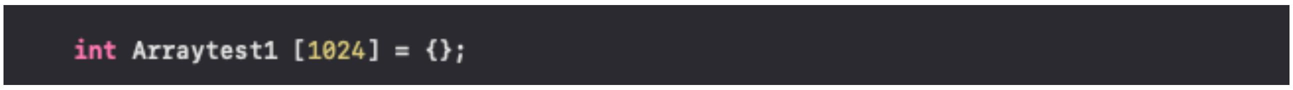
After kernel execution, display the resulting contents on screen.

Kernel (vecadd.cl)



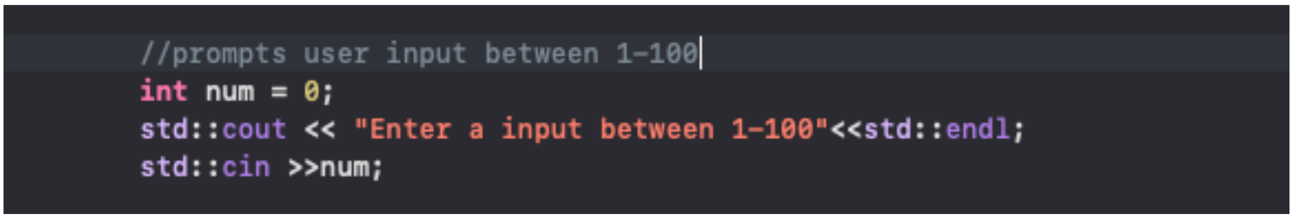
```
__kernel void vecadd(int num, __global int *arr )
    {

        int i = get_global_id(0);
        arr[i] = (i*num) + 1;


    }

```

Creating an array to take in 1024 integers.

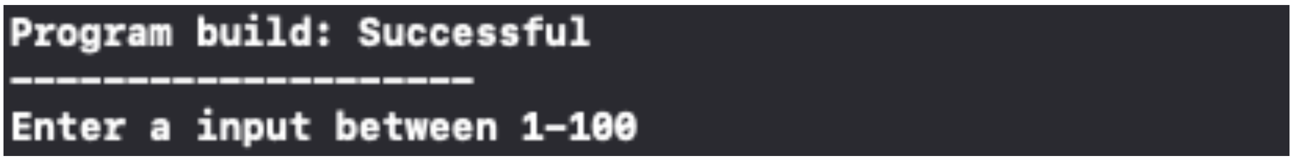


```
int Arraytest1 [1024] = {};
```

Prompts user to enter number between 1-100

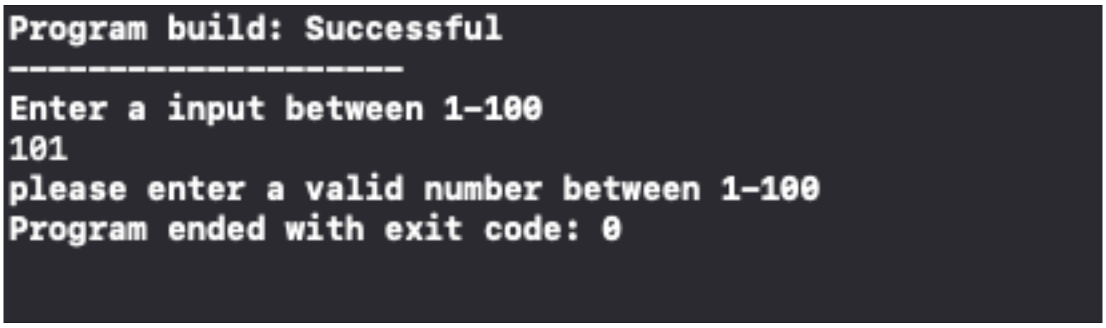


```
//prompts user input between 1-100
int num = 0;
std::cout << "Enter a input between 1-100"<<std::endl;
std::cin >>num;
```

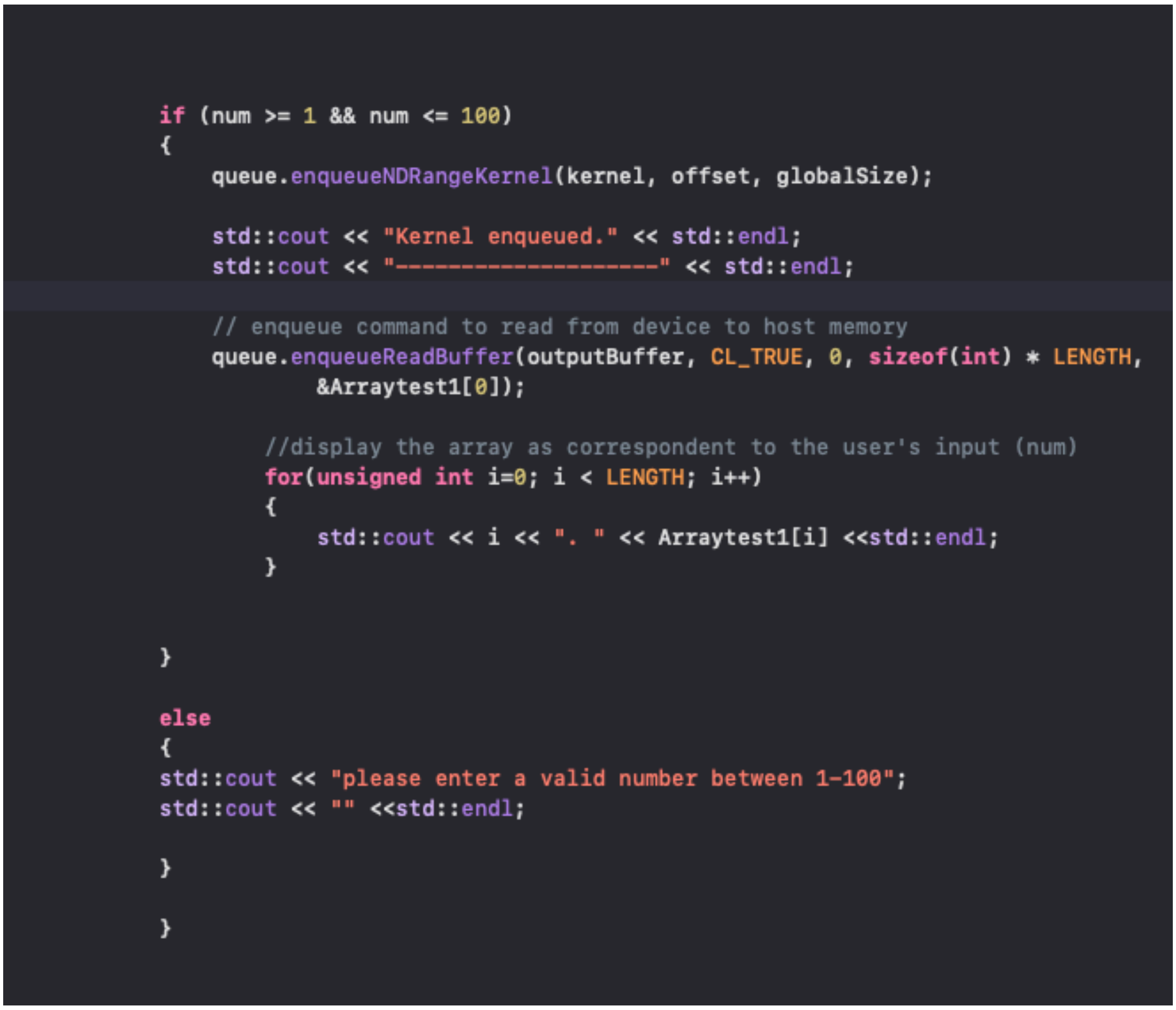


```
Program build: Successful
--------------------
Enter a input between 1-100
```

Program to check whether the user entered a valid number, if not the program will quit

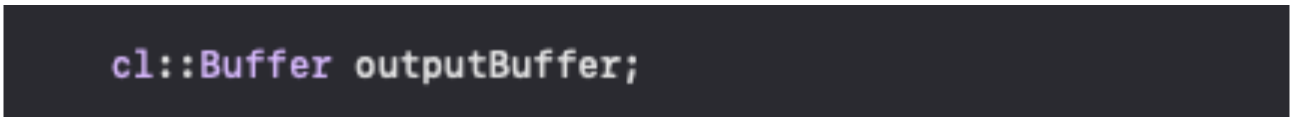


```
Program build: Successful
--------------------
Enter a input between 1-100
101
please enter a valid number between 1-100
Program ended with exit code: 0
```

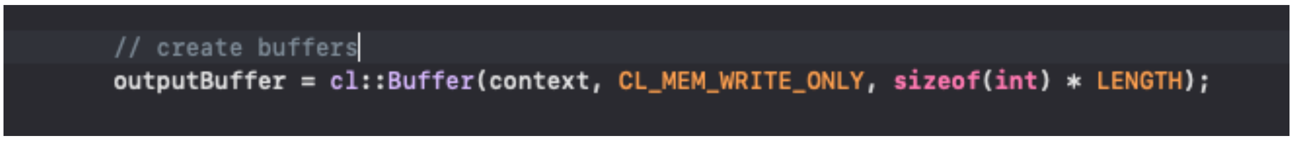


```
if (num >= 1 && num <= 100)
{
    queue.enqueueNDRangeKernel(kernel, offset, globalSize);

    std::cout << "Kernel enqueued." << std::endl;
    std::cout << "--------------------" << std::endl;

    // enqueue command to read from device to host memory
    queue.enqueueReadBuffer(outputBuffer, CL_TRUE, 0, sizeof(int) * LENGTH,
            &Arraytest1[0]);

        //display the array as correspondent to the user's input (num)
        for(unsigned int i=0; i < LENGTH; i++)
        {
            std::cout << i << ". " << Arraytest1[i] <<std::endl;
        }


}

else
{
std::cout << "please enter a valid number between 1-100";
std::cout << "" <<std::endl;

}

}
```

Creating OutputBuffer

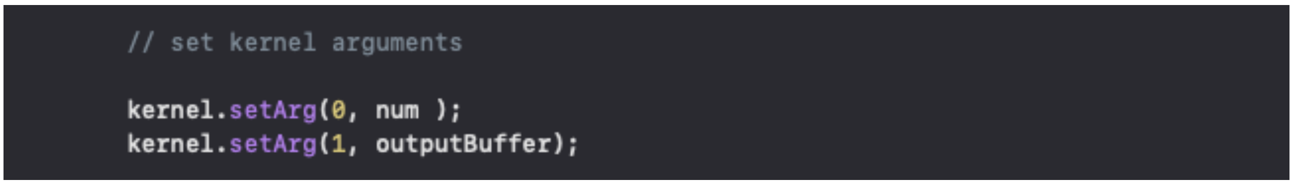


```
cl::Buffer outputBuffer;
```



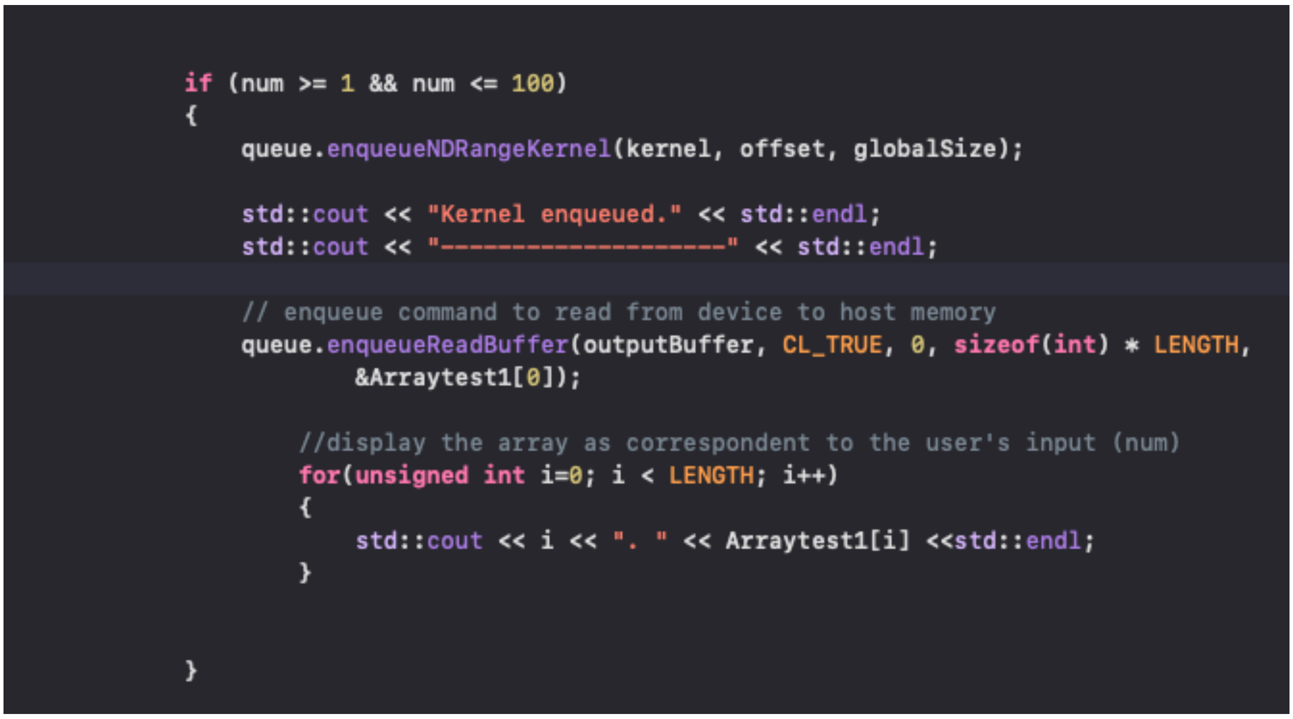
```
// create buffers
outputBuffer = cl::Buffer(context, CL_MEM_WRITE_ONLY, sizeof(int) * LENGTH);
```

Setting kernel arguments. Passing num (user input of 1-100) to kernel & setting the second kernel argument to the outputBuffer.

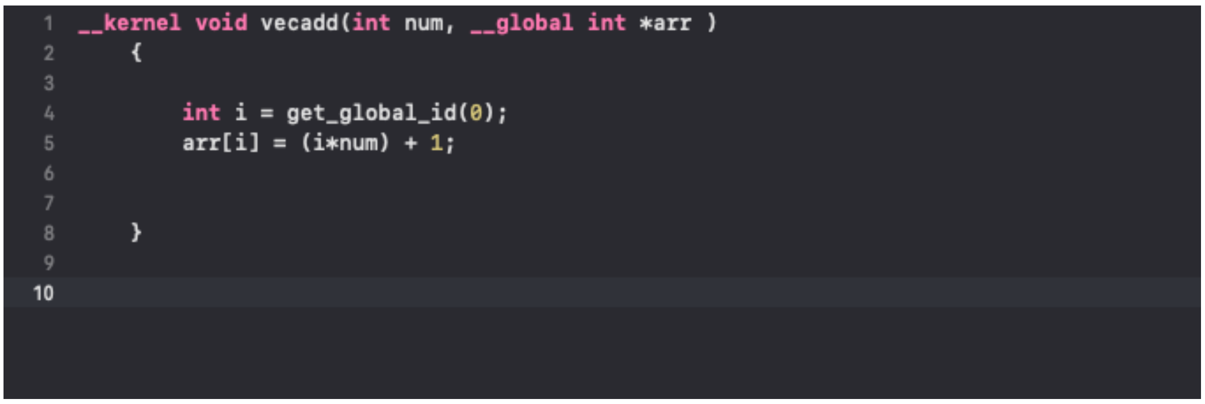


```
// set kernel arguments

kernel.setArg(0, num );
kernel.setArg(1, outputBuffer);
```

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:



```
if (num >= 1 && num <= 100)
{
    queue.enqueueNDRangeKernel(kernel, offset, globalSize);

    std::cout << "Kernel enqueued." << std::endl;
    std::cout << "--------------------" << std::endl;

    // enqueue command to read from device to host memory
    queue.enqueueReadBuffer(outputBuffer, CL_TRUE, 0, sizeof(int) * LENGTH,
            &Arraytest1[0]);

        //display the array as correspondent to the user's input (num)
        for(unsigned int i=0; i < LENGTH; i++)
        {
            std::cout << i << ". " << Arraytest1[i] <<std::endl;
        }


}
```



```
__kernel void vecadd(int num, __global int *arr )
    {

        int i = get_global_id(0);
        arr[i] = (i*num) + 1;


    }


```

<u>If the user enters 1, the resulting contents of the array should be: 1, 2, 3, 4, 5,... until 1024</u>

<u>If the user enters 2, the resulting contents of the array should be: 1, 3, 5, 7, 9,... until 2047</u>

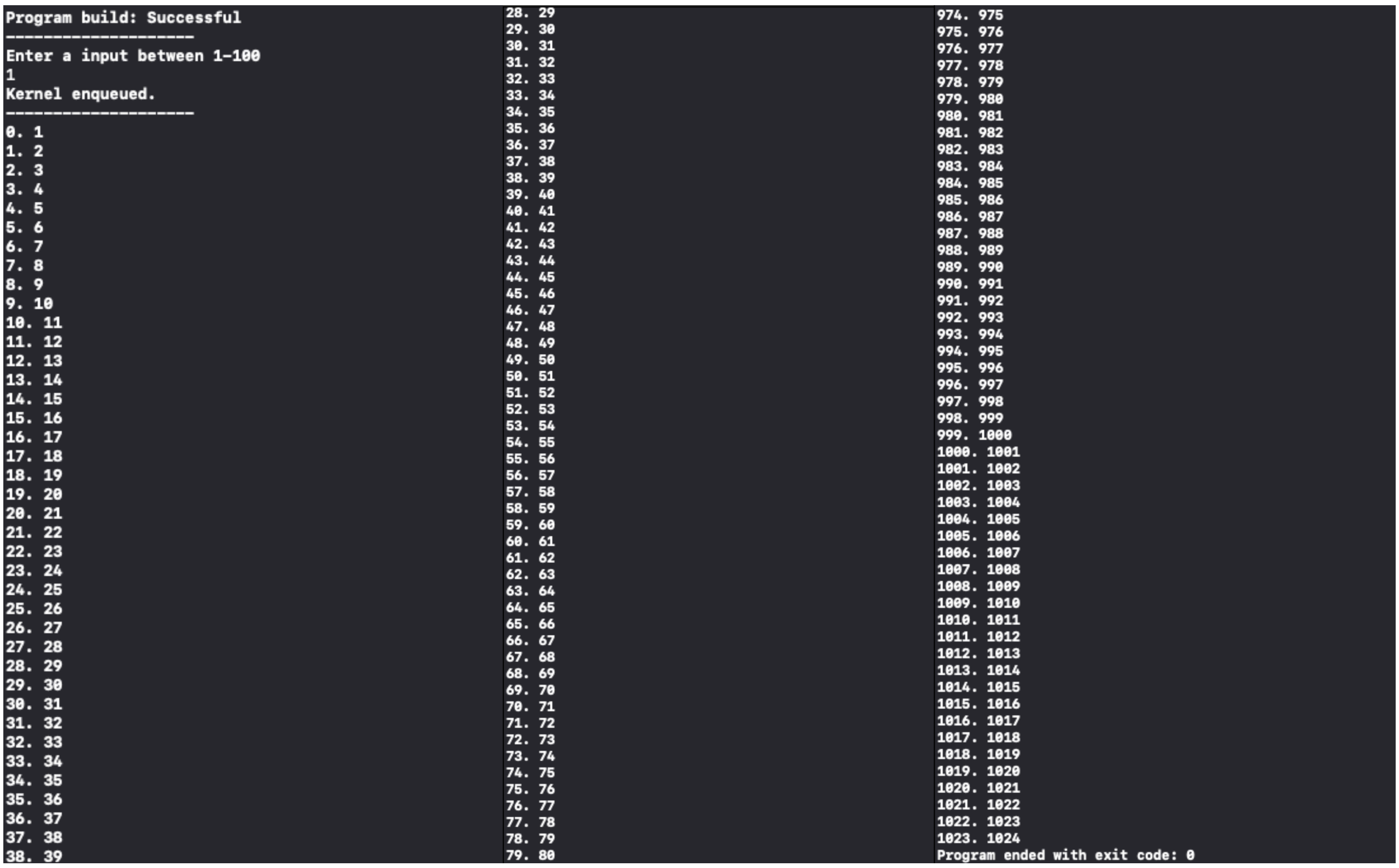
<u>If the user enters 3, the resulting contents of the array should be: 1, 4, 7, 10, 13,... until 3070</u>

<u>If the user enters 100, the resulting contents of the array should be: 1, 101, 201, 301, 401,... until 102301</u>

<u>After kernel execution, display the resulting contents on screen.</u>

<u>Program full run with user inputs 1,2,3 & 100</u>

<u>USER INPUTS 1</u>

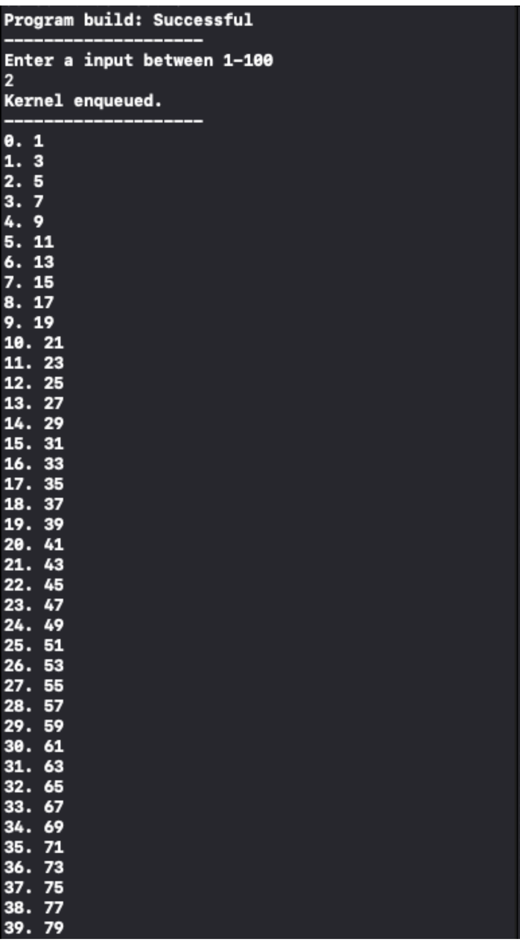


```
Program build: Successful
--------------------
Enter a input between 1-100
1
Kernel enqueued.
--------------------
0. 1
1. 2
2. 3
3. 4
4. 5
5. 6
6. 7
7. 8
8. 9
9. 10
10. 11
11. 12
12. 13
13. 14
14. 15
15. 16
16. 17
17. 18
18. 19
19. 20
20. 21
21. 22
22. 23
23. 24
24. 25
25. 26
26. 27
27. 28
28. 29
29. 30
30. 31
31. 32
32. 33
33. 34
34. 35
35. 36
36. 37
37. 38
38. 39
```

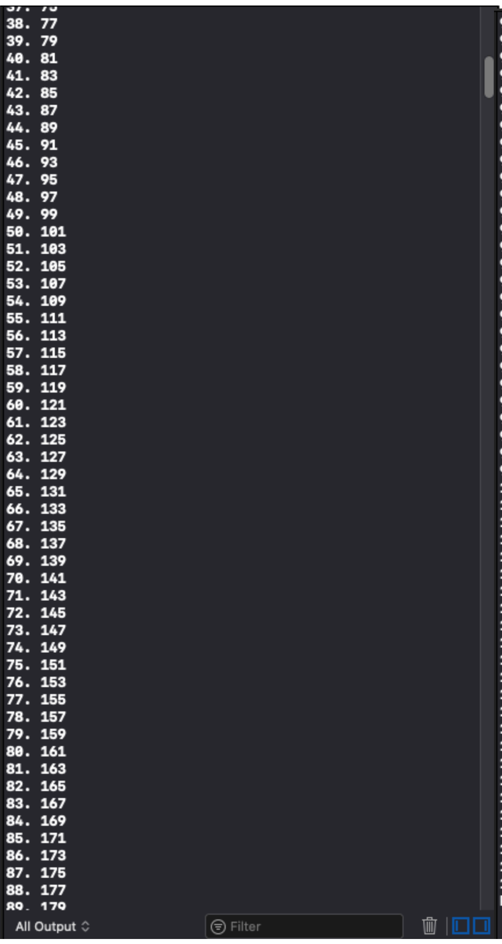
```
28. 29
29. 30
30. 31
31. 32
32. 33
33. 34
34. 35
35. 36
36. 37
37. 38
38. 39
39. 40
40. 41
41. 42
42. 43
43. 44
44. 45
45. 46
46. 47
47. 48
48. 49
49. 50
50. 51
51. 52
52. 53
53. 54
54. 55
55. 56
56. 57
57. 58
58. 59
59. 60
60. 61
61. 62
62. 63
63. 64
64. 65
65. 66
66. 67
67. 68
68. 69
69. 70
70. 71
71. 72
72. 73
73. 74
74. 75
75. 76
76. 77
77. 78
78. 79
79. 80
```

```
974. 975
975. 976
976. 977
977. 978
978. 979
979. 980
980. 981
981. 982
982. 983
983. 984
984. 985
985. 986
986. 987
987. 988
988. 989
989. 990
990. 991
991. 992
992. 993
993. 994
994. 995
995. 996
996. 997
997. 998
998. 999
999. 1000
1000. 1001
1001. 1002
1002. 1003
1003. 1004
1004. 1005
1005. 1006
1006. 1007
1007. 1008
1008. 1009
1009. 1010
1010. 1011
1011. 1012
1012. 1013
1013. 1014
1014. 1015
1015. 1016
1016. 1017
1017. 1018
1018. 1019
1019. 1020
1020. 1021
1021. 1022
1022. 1023
1023. 1024
Program ended with exit code: 0
```

USER INPUTS 2:



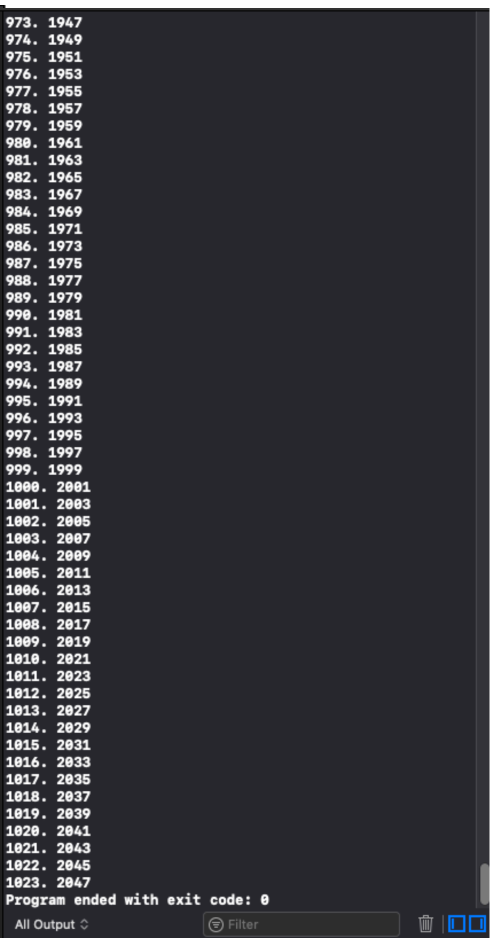
```
Program build: Successful
--------------------
Enter a input between 1-100
2
Kernel enqueued.
--------------------
0. 1
1. 3
2. 5
3. 7
4. 9
5. 11
6. 13
7. 15
8. 17
9. 19
10. 21
11. 23
12. 25
13. 27
14. 29
15. 31
16. 33
17. 35
18. 37
19. 39
20. 41
21. 43
22. 45
23. 47
24. 49
25. 51
26. 53
27. 55
28. 57
29. 59
30. 61
31. 63
32. 65
33. 67
34. 69
35. 71
36. 73
37. 75
38. 77
39. 79
```



```
38. 77
39. 79
40. 81
41. 83
42. 85
43. 87
44. 89
45. 91
46. 93
47. 95
48. 97
49. 99
50. 101
51. 103
52. 105
53. 107
54. 109
55. 111
56. 113
57. 115
58. 117
59. 119
60. 121
61. 123
62. 125
63. 127
64. 129
65. 131
66. 133
67. 135
68. 137
69. 139
70. 141
71. 143
72. 145
73. 147
74. 149
75. 151
76. 153
77. 155
78. 157
79. 159
80. 161
81. 163
82. 165
83. 167
84. 169
85. 171
86. 173
87. 175
88. 177
89. 179
```

All Output

Filter

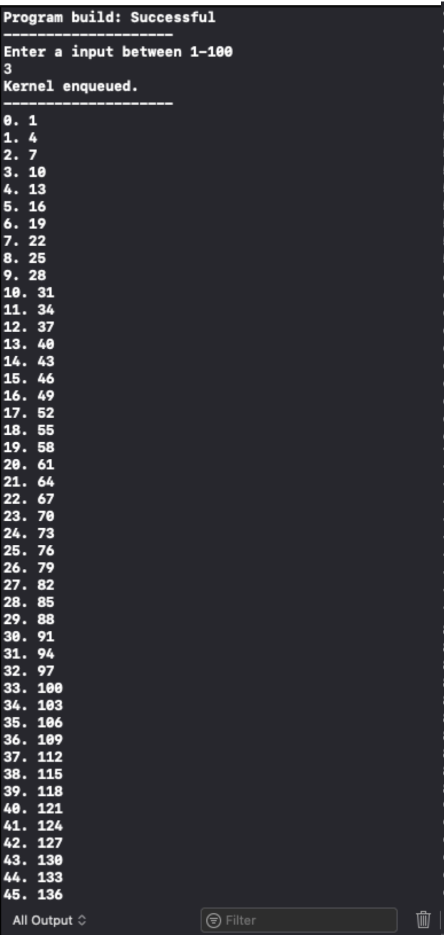


```
973. 1947
974. 1949
975. 1951
976. 1953
977. 1955
978. 1957
979. 1959
980. 1961
981. 1963
982. 1965
983. 1967
984. 1969
985. 1971
986. 1973
987. 1975
988. 1977
989. 1979
990. 1981
991. 1983
992. 1985
993. 1987
994. 1989
995. 1991
996. 1993
997. 1995
998. 1997
999. 1999
1000. 2001
1001. 2003
1002. 2005
1003. 2007
1004. 2009
1005. 2011
1006. 2013
1007. 2015
1008. 2017
1009. 2019
1010. 2021
1011. 2023
1012. 2025
1013. 2027
1014. 2029
1015. 2031
1016. 2033
1017. 2035
1018. 2037
1019. 2039
1020. 2041
1021. 2043
1022. 2045
1023. 2047
Program ended with exit code: 0
```

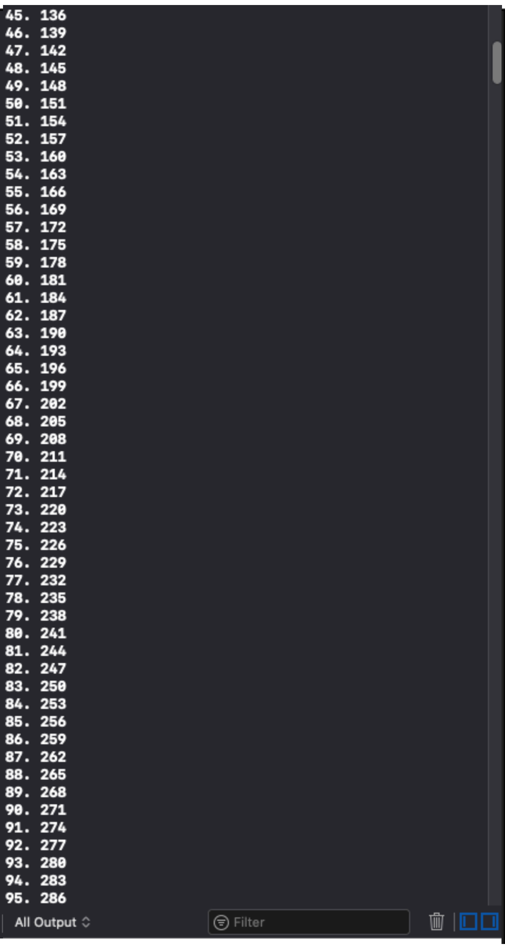
All Output

Filter

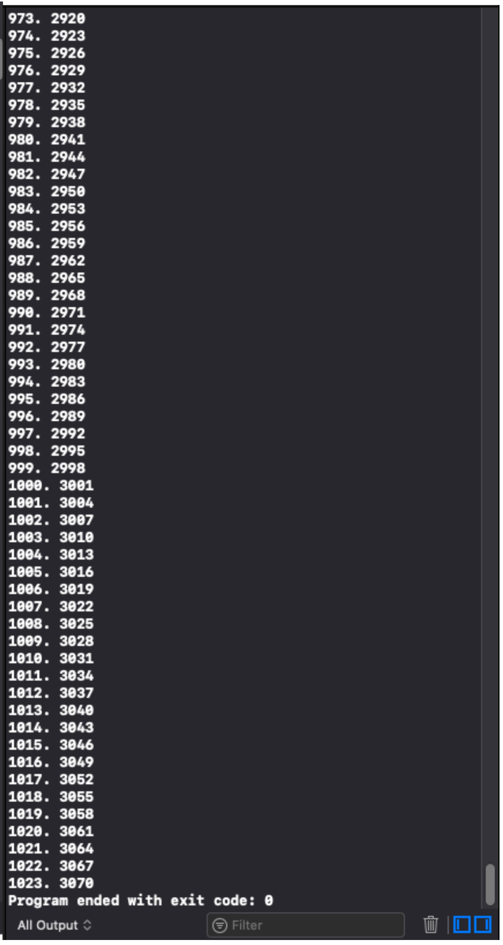
USER INPUTS 3:



```
Program build: Successful
--------------------
Enter a input between 1-100
3
Kernel enqueued.
--------------------
0. 1
1. 4
2. 7
3. 10
4. 13
5. 16
6. 19
7. 22
8. 25
9. 28
10. 31
11. 34
12. 37
13. 40
14. 43
15. 46
16. 49
17. 52
18. 55
19. 58
20. 61
21. 64
22. 67
23. 70
24. 73
25. 76
26. 79
27. 82
28. 85
29. 88
30. 91
31. 94
32. 97
33. 100
34. 103
35. 106
36. 109
37. 112
38. 115
39. 118
40. 121
41. 124
42. 127
43. 130
44. 133
45. 136
```

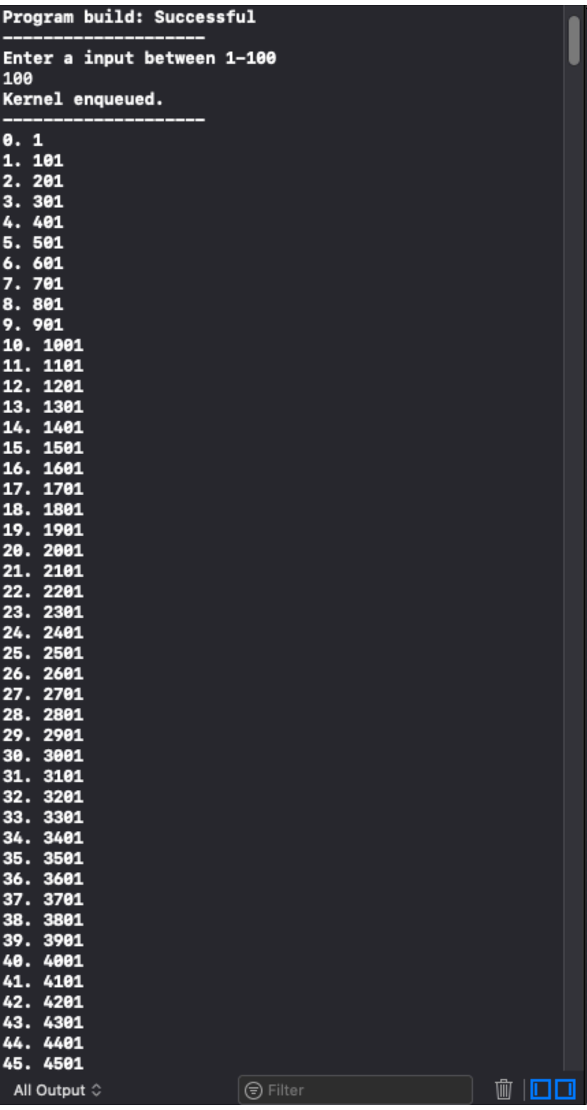


```
45. 136
46. 139
47. 142
48. 145
49. 148
50. 151
51. 154
52. 157
53. 160
54. 163
55. 166
56. 169
57. 172
58. 175
59. 178
60. 181
61. 184
62. 187
63. 190
64. 193
65. 196
66. 199
67. 202
68. 205
69. 208
70. 211
71. 214
72. 217
73. 220
74. 223
75. 226
76. 229
77. 232
78. 235
79. 238
80. 241
81. 244
82. 247
83. 250
84. 253
85. 256
86. 259
87. 262
88. 265
89. 268
90. 271
91. 274
92. 277
93. 280
94. 283
95. 286
```

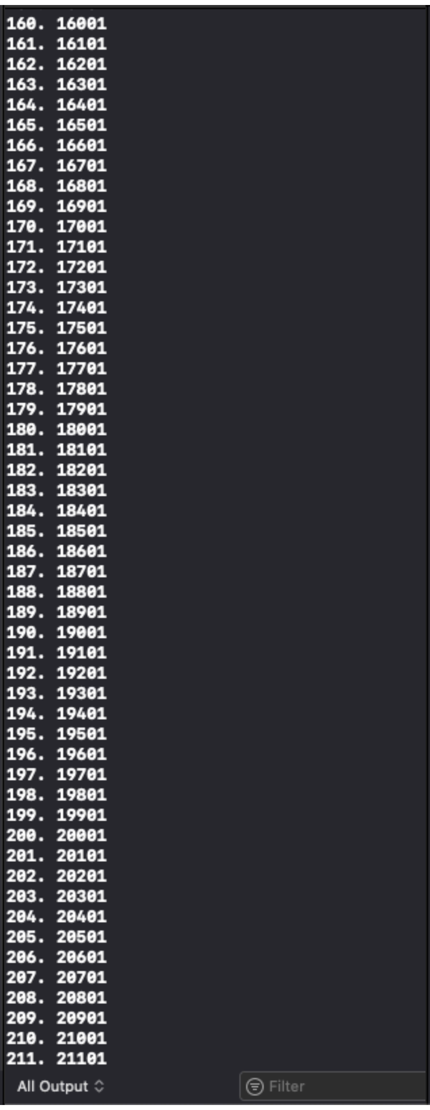


```
973. 2920
974. 2923
975. 2926
976. 2929
977. 2932
978. 2935
979. 2938
980. 2941
981. 2944
982. 2947
983. 2950
984. 2953
985. 2956
986. 2959
987. 2962
988. 2965
989. 2968
990. 2971
991. 2974
992. 2977
993. 2980
994. 2983
995. 2986
996. 2989
997. 2992
998. 2995
999. 2998
1000. 3001
1001. 3004
1002. 3007
1003. 3010
1004. 3013
1005. 3016
1006. 3019
1007. 3022
1008. 3025
1009. 3028
1010. 3031
1011. 3034
1012. 3037
1013. 3040
1014. 3043
1015. 3046
1016. 3049
1017. 3052
1018. 3055
1019. 3058
1020. 3061
1021. 3064
1022. 3067
1023. 3070
Program ended with exit code: 0
```

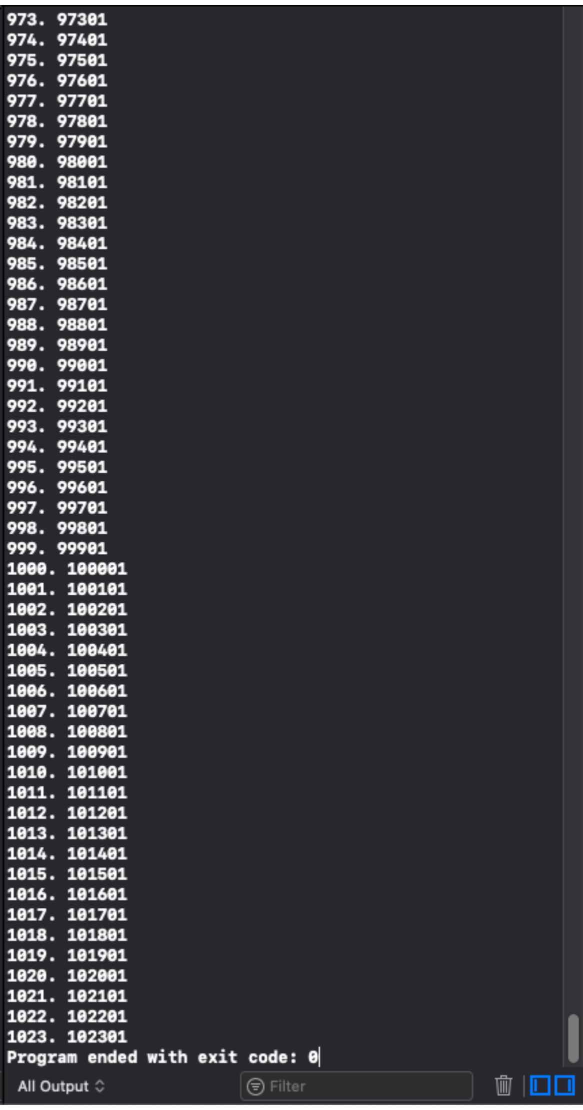
<u>User Inputs 100 (largest possible user input):</u>



```
Program build: Successful
--------------------
Enter a input between 1-100
100
Kernel enqueued.
--------------------
0. 1
1. 101
2. 201
3. 301
4. 401
5. 501
6. 601
7. 701
8. 801
9. 901
10. 1001
11. 1101
12. 1201
13. 1301
14. 1401
15. 1501
16. 1601
17. 1701
18. 1801
19. 1901
20. 2001
21. 2101
22. 2201
23. 2301
24. 2401
25. 2501
26. 2601
27. 2701
28. 2801
29. 2901
30. 3001
31. 3101
32. 3201
33. 3301
34. 3401
35. 3501
36. 3601
37. 3701
38. 3801
39. 3901
40. 4001
41. 4101
42. 4201
43. 4301
44. 4401
45. 4501
```



```
160. 16001
161. 16101
162. 16201
163. 16301
164. 16401
165. 16501
166. 16601
167. 16701
168. 16801
169. 16901
170. 17001
171. 17101
172. 17201
173. 17301
174. 17401
175. 17501
176. 17601
177. 17701
178. 17801
179. 17901
180. 18001
181. 18101
182. 18201
183. 18301
184. 18401
185. 18501
186. 18601
187. 18701
188. 18801
189. 18901
190. 19001
191. 19101
192. 19201
193. 19301
194. 19401
195. 19501
196. 19601
197. 19701
198. 19801
199. 19901
200. 20001
201. 20101
202. 20201
203. 20301
204. 20401
205. 20501
206. 20601
207. 20701
208. 20801
209. 20901
210. 21001
211. 21101
```

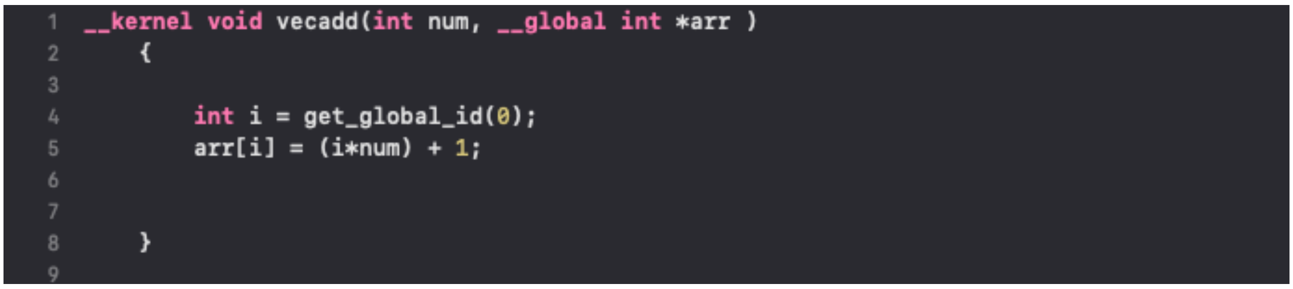


```
973. 97301
974. 97401
975. 97501
976. 97601
977. 97701
978. 97801
979. 97901
980. 98001
981. 98101
982. 98201
983. 98301
984. 98401
985. 98501
986. 98601
987. 98701
988. 98801
989. 98901
990. 99001
991. 99101
992. 99201
993. 99301
994. 99401
995. 99501
996. 99601
997. 99701
998. 99801
999. 99901
1000. 100001
1001. 100101
1002. 100201
1003. 100301
1004. 100401
1005. 100501
1006. 100601
1007. 100701
1008. 100801
1009. 100901
1010. 101001
1011. 101101
1012. 101201
1013. 101301
1014. 101401
1015. 101501
1016. 101601
1017. 101701
1018. 101801
1019. 101901
1020. 102001
1021. 102101
1022. 102201
1023. 102301
Program ended with exit code: 0
```

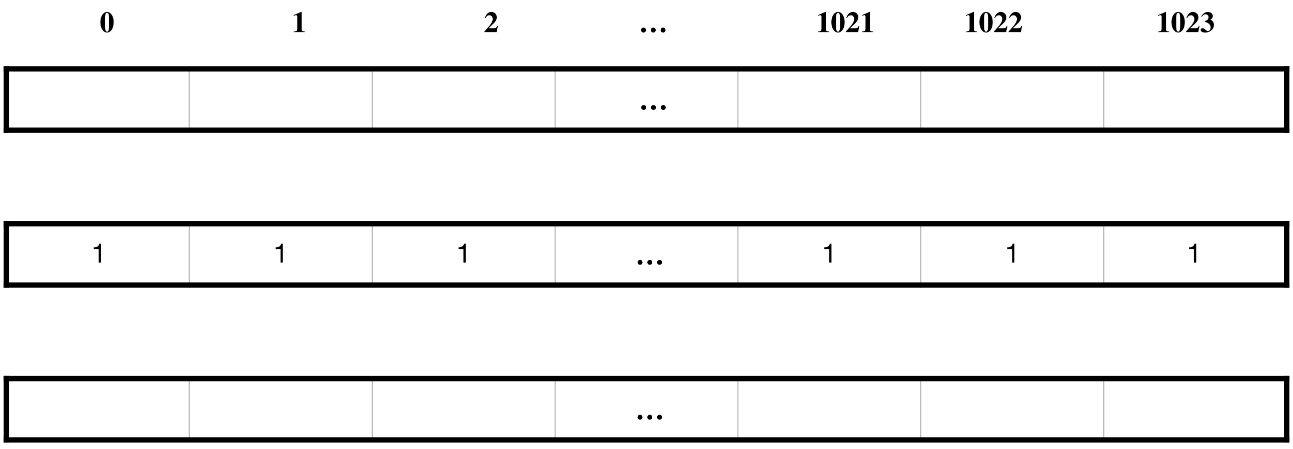
TASK 3 DESCRIPTION

Explanation & Diagram:

The below diagram and explanation depicts the process in the kernel running in parallel.



```
__kernel void vecadd(int num, __global int *arr )
    {

        int i = get_global_id(0);
        arr[i] = (i*num) + 1;


    }
```



Num = user input (1-100)

*—> (i*num) + 1 = arr[i]*

Example:

Num (user input) = 50

*arr[0] = (50 * 0) + 1 = 1*

*arr[1] = (50 * 1) + 1 = 51*

*arr[2] = (50 * 2) + 1 = 101*

…

*arr[1023] = (50 * 1023) + 1 = 51151*