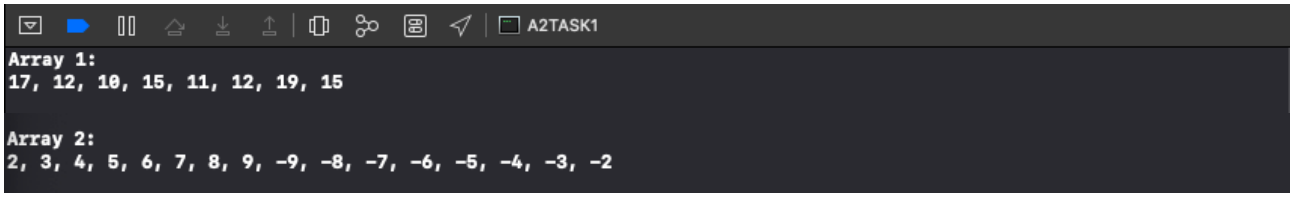
Assignment 1 report

Name: Yeo Zheng Xu Isaac

TASK1 - OPENCL Vector DataType

- Write a program that does the following:
  In the host, create two arrays as follows:

o Array 1: An 8 element array of ints with random values between 10 and 20.
o Array 2: A 16 element array of ints. Initialise the first half of the array with values from 2 to 9 and the second half with values from -9 to -2.




```
    //initialise arrays

    unsigned int array1[8];

    srand((unsigned)time(0));

    std::cout << "Array 1:" << std::endl;
    for (int i = 0; i < 8; i++)
    {
        if (i == 7)
        {
            array1[i] = (rand() % (21 - 10) + 10);
            std::cout << array1[i];
        }
        else
        {
            array1[i] = (rand() % (21 - 10) + 10);
            std::cout << array1[i] << ", ";
        }
    }


    int array2[16] = { 2,3,4,5,6,7,8,9,-9,-8,-7,-6,-5,-4,-3,-2 };

    std::cout << std::endl;
    std::cout << std::endl;
    std::cout << "Array 2:" << std::endl;
    for (int i = 0; i < 16; i++)
    {
        if (i == 15)
        {
            std::cout << array2[i];
        }
        else
        {
            std::cout << array2[i] << ", ";
        }
    }
    std::cout << std::endl;
    std::cout << std::endl;
```

- Write a kernel that
  - Accepts array 1 as an array of int4s, array 2 and an output array
  - Reads the contents from array 1 and 2 into local memory
  - Copy the contents of array 1 into an int8 vector called *v*
  - Copy the contents (using **vload*n***) of array 2 into two int8 vectors called *v1* and *v2*

//v1 = vload8(0,arr2); v2=vload8(1,arr2);

```
__kernel void part1 (global int4 *arr1, global int *arr2, global int *out)
{
    local int8 v;
    v.lo=arr1[0];
    v.hi=arr1[1];

    local int8 v1,v2;
    v1=vload8(0,arr2);
    v2=vload8(1,arr2);

    int8 results;
    int8 ref={15,15,15,15,15,15,15,15};
    int8 cmp= ref-v;

    if(any(cmp))
    {
        //some element is >15
        results=select(v2,v1,cmp);
    }
    else
    {
        results.lo=v1.lo;
        results.hi=v2.lo;
    }

    vstore8(v,0,out);
    vstore8(v1,1,out);
    vstore8(v2,2,out);
    vstore8(results,3,out);

}

```

- Creates an int8 vector in private memory called *results*. The contents of this vector should be filled as follows:
  - Check whether **any** of the elements in *v* are greater than 15

  - If there are, then for elements that are greater than 15, copy the corresponding elements from *v1* into *results*; for elements less than or equal to 15, copy the elements from *v2* into *results*. (Use **select**).

  - If not, fill the first 4 elements of *results* with the contents from the first 4 elements of *v1*; and fill the next 4 elements of *results* with contents from the first 4 elements of *v2*.

```
int v[8];
int v1[8];
int v2[8];

for (int i = 0; i < 8; i++)
{
    v[i] = array1[i];
}

for (int i = 0; i < 8; i++)
{
    v1[i] = array2[i];
}

for (int i = 0; i < 8; i++)
{
    v2[i] = array2[8 + i];
}


int results[32];
for (int i = 0; i < 8; i++)
{
    results[i] = v[i];
}
for (int i = 0; i < 8; i++)
{
    results[8 + i] = v1[i];
}
for (int i = 0; i < 8; i++)
{
    results[16 + i] = v2[i];
}


int above_15 = false;

for (int i = 0; i < 8; i++)
{
    if (v[i] > 15)
    {
        above_15 = true;
    }
}

//if element in v > 15, copy corresponding element from v1
if (above_15)
{
    for (int i = 0; i < 8; i++)
    {
        if (v[i] > 15)
        {
            results[24 + i] = v1[i];
        }
        else
        {
            results[24 + i] = v2[i];
        }
     }

 }
 //if element in v < 15, copy corresponding element from v2
 else
 {
     for (int i = 0; i < 4; i++)
     {
         results[24 + i] = v1[i];
     }

     for (int i = 0; i < 4; i++)
     {
         results[24 + 4 + i] = v2[i];
     }
 }
```

```
//KERNEL
    int8 results;
    int8 ref={15,15,15,15,15,15,15,15};
    int8 cmp= ref-v;

    if(any(cmp))
    {
        //some element is >15
        results=select(v2,v1,cmp);
    }
    else
    {
        results.lo=v1.lo;
        results.hi=v2.lo;
    }
```

- Stores the contents of *v*, *v1*, *v2* and *results* in the output array (using **vstore*n***) Note that the host will only have to enqueue 1 work item.

```
    vstore8(v,0,out);
    vstore8(v1,1,out);
    vstore8(v2,2,out);
    vstore8(results,3,out);
```

```
__kernel void part1 (global int4 *arr1, global int *arr2, global int *out)
{
    local int8 v;
    v.lo=arr1[0];
    v.hi=arr1[1];

    local int8 v1,v2;
    v1=vload8(0,arr2);
    v2=vload8(1,arr2);

    int8 results;
    int8 ref={15,15,15,15,15,15,15,15};
    int8 cmp= ref-v;

    if(any(cmp))
    {
        //some element is >15
        results=select(v2,v1,cmp);
    }
    else
    {
        results.lo=v1.lo;
        results.hi=v2.lo;
    }

    vstore8(v,0,out);
    vstore8(v1,1,out);
    vstore8(v2,2,out);
    vstore8(results,3,out);

}
```

```
    //make buffers
    cl_mem buffer_array1 = clCreateBuffer(context, CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR,
        sizeof(array1), array1, &err);
    cl_mem buffer_array2 = clCreateBuffer(context, CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR,
        sizeof(array2), array2, &err);
    cl_mem buffer_output = clCreateBuffer(context, CL_MEM_READ_ONLY, 32 * sizeof(int), NULL, &err);//32
        bytes for results

    size_t workgroups[] = { 1,0,0 };


    err = clSetKernelArg(kernel, 0, sizeof(buffer_array1), &buffer_array1);
    err = clSetKernelArg(kernel, 1, sizeof(buffer_array2), &buffer_array2);
    err = clSetKernelArg(kernel, 2, sizeof(buffer_output), &buffer_output);

    err = clEnqueueNDRangeKernel(queue, kernel, 1, NULL,
        workgroups, NULL, 0, NULL, NULL);


    int outputArray[32] = {};

    ///read results
    clEnqueueReadBuffer(queue, buffer_output, true, 0, sizeof(outputArray), outputArray, 0, NULL, NULL);


    //wait for copy
    clFinish(queue);
```

```
    //output results
    std::cout << "Output Array:" << std::endl;
    for (int k = 0; k < 32; k += 8)
    {
        for (int i = 0; i < 8; i++)
        {
            printf("%3d", results[k + i]);
        }

        std::cout << std::endl;
    }
    std::cout << std::endl;

    getchar();

    return 0;
}
```

//Output

```
Output Array:
 17 18 19 13 12 14 13 14
  2  3  4  5  6  7  8  9
 -9 -8 -7 -6 -5 -4 -3 -2
  2  3  4 -6 -5 -4 -3 -2
```

- In the host, check that the results are correct and display the contents of the output array.

```
Array 1:
17, 18, 19, 13, 12, 14, 13, 14

Array 2:
2, 3, 4, 5, 6, 7, 8, 9, -9, -8, -7, -6, -5, -4, -3, -2

Output Array:
 17 18 19 13 12 14 13 14
  2  3  4  5  6  7  8  9
 -9 -8 -7 -6 -5 -4 -3 -2
  2  3  4 -6 -5 -4 -3 -2
```

If not, fill the first 4 elements of results with the contents from the first 4 elements of v1; and fill the next 4 elements of results with contents from the first 4 elements of v2.

2 3 4 5 -9 -8 -7 -6

```
Array 1:
11, 10, 12, 12, 12, 11, 10, 13

Array 2:
2, 3, 4, 5, 6, 7, 8, 9, -9, -8, -7, -6, -5, -4, -3, -2

Output Array:
 11 10 12 12 12 11 10 13
  2  3  4  5  6  7  8  9
 -9 -8 -7 -6 -5 -4 -3 -2
  2  3  4  5 -9 -8 -7 -6
```

Task 2 – Shift Cipher

Leave anything that is not an alphabet as is (i.e. punctuations and spaces). Decrypting the ciphertext is simply a matter of reversing the shift.

Task 2a

Write a normal C/C++ program (not using OpenCL) that reads the contents from a text file called “plaintext.txt” (a test file has been provided). The program should prompt the user to input a valid *n* value, then encrypt the plaintext using the shift cipher method described above, and output the ciphertext into an output text file called “ciphertext.txt”. To ensure that the encryption was performed correctly, your program must also decrypt the ciphertext into a file called “decrypted.txt” to check whether it matches the original plaintext (albeit in upper case).

1. LOAD PLAINTEXT FILE INTO PROGRAM

```
int main(int argc, const char * argv[]) {
    std::cout << "-- Loading plaintext.txt --" << std::endl;

    std::vector<char> fileChar;
    fileChar = returnFileContents("plaintext.txt");
```

2. Program menu for user’s input (1. Normal Encryption/Decryption)

```
    std::cout << "Name: Yeo Zheng Xu Isaac" << std::endl;
    std::cout << "Student ID: 6342425" << std::endl;
    std::cout << "-----------------------------------" << std::endl;
    std::cout << "\t 1. -- 2a. Run using C++ program" << std::endl;
    std::cout << "\t 2. -- 2b. Run using OpenCL program" << std::endl;
    std::cout << "\t 3. -- 2c. Run OpenCL Lookup table program" << std::endl;

    int input = getRangeInput(1,3, "Select an option");
    int shift;

    if(input == 1){
         shift = getInput("Select a shift amount");
        encrypt(fileChar,shift);
```

3.

Encryption function to encrypt the data read into the program from plaintext.txt, then appending each encrypted cipher text into the txt file named "output.txt", if spaces or symbols are detected, append that respective symbol or space into the txt file.

```
    std::cout << "\nEncrypting into output.txt " << std::endl;
    std::ofstream file;
    file.open ("output.txt", std::ios_base::app);

    for(it = loadedString.begin(); it < loadedString.end(); it++){
        char getChar = *it;
         if(getChar != ' '){    // if is char
             if(getChar < 'A'){
                encryptedString.push_back(getChar);
                 file << getChar;

            }
             else{
                 char encryptedChar = ((getChar+encryptedBuffer-65)%26+'A');
                 encryptedString.push_back(encryptedChar);
                 file << encryptedChar;
             }
        }
         else{ // append space if space
             encryptedString.push_back(' ');
             file << ' ';
        }
    }
    file.close();
    std::cout << "Done " << std::endl;
```

4.

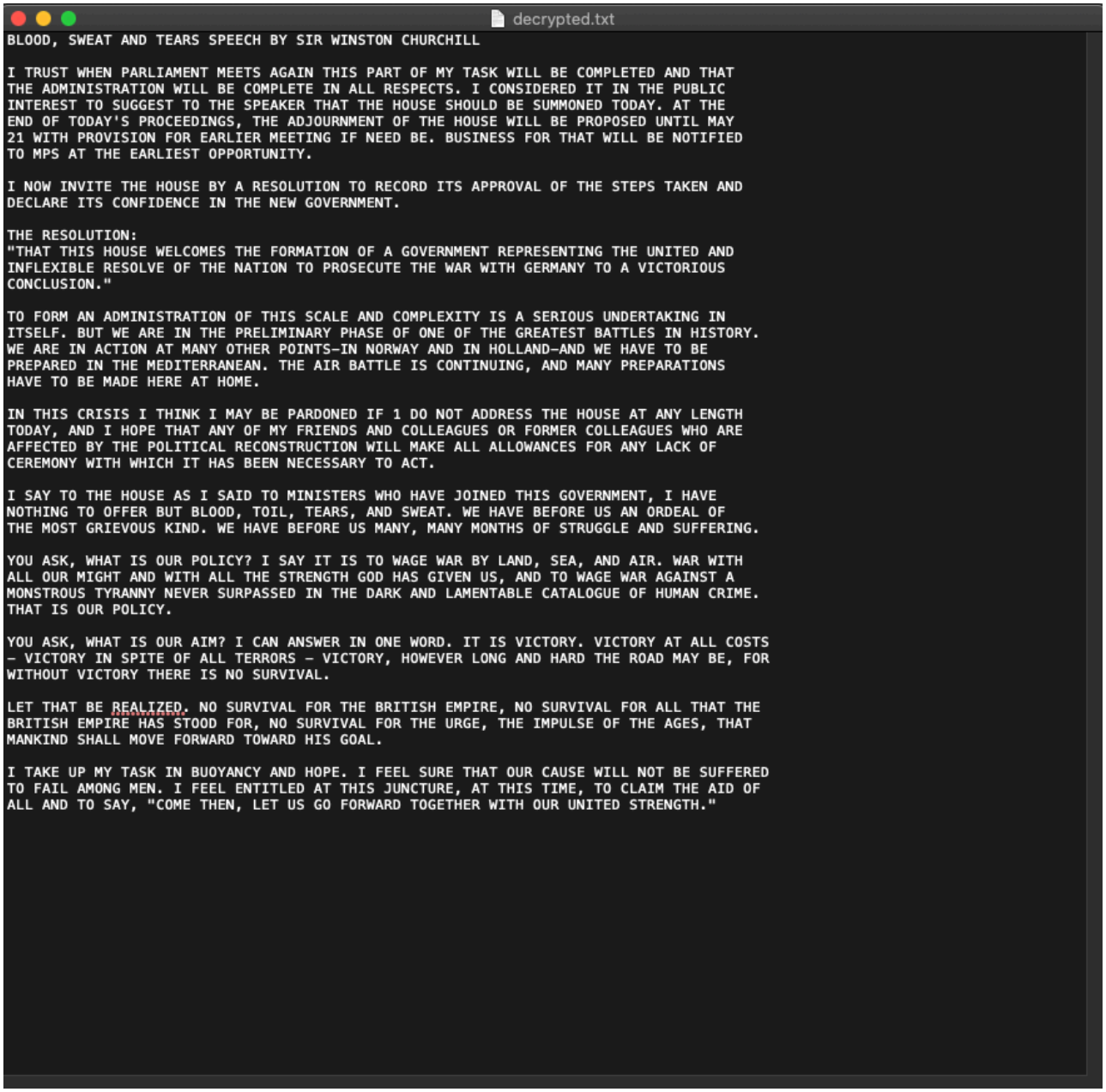
Decryption function to decrypt the current encrypted string back into plaintext then appending each decrypted plaintext alphabet into the txt file named "decrypted.txt", if spaces or symbols are detected, append that respective symbol or space into the txt file.

```
    std::cout << std::endl;
    std::cout << "-- Decrypting into decrypted.txt --" << std::endl;
    // decryption.
    file.open ("decrypted.txt", std::ios_base::app);
    for(it = encryptedString.begin(); it < encryptedString.end(); it++){
        char charact = *it;
         if(charact != ' '){
             if ((charact < 'A') || (charact > 'Z')){
                 decryptedString.push_back(charact);
                 file << charact;
             }
             else{
                char plainChar = ((charact - encryptedBuffer-65)%26)+'A';
                if(plainChar < 'A')
                    plainChar = 'Z'- ('A'-plainChar-1);
                if(plainChar > 'Z')
                    plainChar = 'Z'-('A'-plainChar-1);
                decryptedString.push_back(plainChar);
                file << plainChar;
             }
        }else{
             decryptedString.push_back(' ');
             file << ' ';
        }
    }
    file.close();
    std::cout << "Done " << std::endl;

}
```

```
-- Loading plaintext.txt --
plaintext.txt loaded
Name: Yeo Zheng Xu Isaac
Student ID: 6342425
---------------------------------------
     1. -- 2a. Run using C++ program
     2. -- 2b. Run using OpenCL program
     3. -- 2c. Run OpenCL Lookup table program
Select an option: 1
Select a shift amount: 5

Encrypting into output.txt
Done

-- Decrypting into decrypted.txt --
Done
Program ended with exit code: 0
```

output.txt

```
GQTTI, XBJFY FSI YJFWX XUJJHM GD XNW BNSXYTS HMZWHMNQQ

N YWZXY BMJS UFWQNFRJSY RJJYX FLFNS YMNX UFWY TK RD YFXP BNQQ GJ HTRUQJYJI FSI YMFY
YMJ FIRNSNXYWFYNTS BNQQ GJ HTRUQJYJ NS FQQ WJXUJHYX. N HTSXNIJWJI NY NS YMJ UZGQNH
NSYJWJXY YT XZLLJXY YT YMJ XUJFPJW YMFY YMJ MTZXJ XMTZQI GJ XZRRTSJI YTIFD. FY YMJ
JSI TK YTIFD'X UWTHJJINSLX, YMJ FIOTZWSRJSY TK YMJ MTZXJ BNQQ GJ UWTUTXJI ZSYNQ RFD
21 BNYM UWTANXNTS KTW JFWQNJW RJJYNSL NK SJJI GJ. GZXNSJXX KTW YMFY BNQQ GJ STYNKNJI
YT RUX FY YMJ JFWQNJXY TUUTWYZSNYD.

N STB NSANYJ YMJ MTZXJ GD F WJXTQZYNTS YT WJHTWI NYX FUUWTAFQ TK YMJ XYJUX YFPJS FSI
IJHQFWJ NYX HTSKNIJSHJ NS YMJ SJB LTAJWSRJSY.

YMJ WJXTQZYNTS:
"YMFY YMNX MTZXJ BJQHTRJX YMJ KTWRFYNTS TK F LTAJWSRJSY WJUWJXJSYNSL YMJ ZSNYJI FSI
NSKQJCNGQJ WJXTQAJ TK YMJ SFYNTS YT UWTXJHZYJ YMJ BFW BNYM LJWRFSD YT F ANHYTWNTZX
HTSHQZXNTS."

YT KTWR FS FIRNSNXYWFYNTS TK YMNX XHFQJ FSI HTRUQJCNYD NX F XJWNTZX ZSIJWYFPNSL NS
NYXJQK. GZY BJ FWJ NS YMJ UWJQNRNSFWD UMFXJ TK TSJ TK YMJ LWJFYJXY GFYYQJX NS MNXYTWD.
BJ FWJ NS FHYNTS FY RFSD TYMJW UTNSYX-NS STWBFD FSI NS MTQQFSI-FSI BJ MFAJ YT GJ
UWJUFWJI NS YMJ RJINYJWWFSJFS. YMJ FNW GFYYQJ NX HTSYNSZNSL, FSI RFSD UWJUFWFYNTSX
MFAJ YT GJ RFIJ MJWJ FY MTRJ.

NS YMNX HWNXNX N YMNSP N RFD GJ UFWITSJI NK 1 IT STY FIIWJXX YMJ MTZXJ FY FSD QJSLYM
YTIFD, FSI N MTUJ YMFY FSD TK RD KWNJSIX FSI HTQQJFLZJX TW KTWRJW HTQQJFLZJX BMT FWJ
FKKJHYJI GD YMJ UTQNYNHFQ WJHTSXYWZHYNTS BNQQ RFPJ FQQ FQQTBFSHJX KTW FSD QFHP TK
HJWJRTSD BNYM BMNHM NY MFX GJJS SJHJXXFWD YT FHY.

N XFD YT YMJ MTZXJ FX N XFNI YT RNSNXYJWX BMT MFAJ OTNSJI YMNX LTAJWSRJSY, N MFAJ
STYMNSL YT TKKJW GZY GQTTI, YTNQ, YJFWX, FSI XBJFY. BJ MFAJ GJKTWJ ZX FS TWIJFQ TK
YMJ RTXY LWNJATZX PNSI. BJ MFAJ GJKTWJ ZX RFSD, RFSD RTSYMX TK XYWZLLQJ FSI XZKKJWNSL.

DTZ FXP, BMFY NX TZW UTQNHD? N XFD NY NX YT BFLJ BFW GD QFSI, XJF, FSI FNW. BFW BNYM
FQQ TZW RNLMY FSI BNYM FQQ YMJ XYWJSLYM LTI MFX LNAJS ZX, FSI YT BFLJ BFW FLFNSXY F
RTSXYWTZX YDWFSSD SJAJW XZWUFXXJI NS YMJ IFWP FSI QFRJSYFGQJ HFYFQTLZJ TK MZRFS HWNRJ.
YMFY NX TZW UTQNHD.

DTZ FXP, BMFY NX TZW FNR? N HFS FSXBJW NS TSJ BTWI. NY NX ANHYTWD. ANHYTWD FY FQQ HTXYX
- ANHYTWD NS XUNYJ TK FQQ YJWWTWX - ANHYTWD, MTBJAJW QTSL FSI MFWI YMJ WTFI RFD GJ, KTW
BNYMTZY ANHYTWD YMJWJ NX ST XZWANAFQ.

QJY YMFY GJ WJFQNEJI. ST XZWANAFQ KTW YMJ GWNYNXM JRUNWJ, ST XZWANAFQ KTW FQQ YMFY YMJ
GWNYNXM JRUNWJ MFX XYTTI KTW, ST XZWANAFQ KTW YMJ ZWLJ, YMJ NRUZQXJ TK YMJ FLJX, YMFY
RFSPNSI XMFQQ RTAJ KTWBFWI YTBFWI MNX LTFQ.

N YFPJ ZU RD YFXP NS GZTDFSHD FSI MTUJ. N KJJQ XZWJ YMFY TZW HFZXJ BNQQ STY GJ XZKKJWJI
YT KFNQ FRTSL RJS. N KJJQ JSYNYQJI FY YMNX OZSHYZWJ, FY YMNX YNRJ, YT HQFNR YMJ FNI TK
FQQ FSI YT XFD, "HTRJ YMJS, QJY ZX LT KTWBFWI YTLJYMJW BNYM TZW ZSNYJI XYWJSLYM."
```


decrypted.txt
BLOOD, SWEAT AND TEARS SPEECH BY SIR WINSTON CHURCHILL
I TRUST WHEN PARLIAMENT MEETS AGAIN THIS PART OF MY TASK WILL BE COMPLETED AND THAT THE ADMINISTRATION WILL BE COMPLETE IN ALL RESPECTS. I CONSIDERED IT IN THE PUBLIC INTEREST TO SUGGEST TO THE SPEAKER THAT THE HOUSE SHOULD BE SUMMONED TODAY. AT THE END OF TODAY'S PROCEEDINGS, THE ADJOURNMENT OF THE HOUSE WILL BE PROPOSED UNTIL MAY 21 WITH PROVISION FOR EARLIER MEETING IF NEED BE. BUSINESS FOR THAT WILL BE NOTIFIED TO MPS AT THE EARLIEST OPPORTUNITY.
I NOW INVITE THE HOUSE BY A RESOLUTION TO RECORD ITS APPROVAL OF THE STEPS TAKEN AND DECLARE ITS CONFIDENCE IN THE NEW GOVERNMENT.
THE RESOLUTION:
"THAT THIS HOUSE WELCOMES THE FORMATION OF A GOVERNMENT REPRESENTING THE UNITED AND INFLEXIBLE RESOLVE OF THE NATION TO PROSECUTE THE WAR WITH GERMANY TO A VICTORIOUS CONCLUSION."
TO FORM AN ADMINISTRATION OF THIS SCALE AND COMPLEXITY IS A SERIOUS UNDERTAKING IN ITSELF. BUT WE ARE IN THE PRELIMINARY PHASE OF ONE OF THE GREATEST BATTLES IN HISTORY. WE ARE IN ACTION AT MANY OTHER POINTS—IN NORWAY AND IN HOLLAND—AND WE HAVE TO BE PREPARED IN THE MEDITERRANEAN. THE AIR BATTLE IS CONTINUING, AND MANY PREPARATIONS HAVE TO BE MADE HERE AT HOME.
IN THIS CRISIS I THINK I MAY BE PARDONED IF 1 DO NOT ADDRESS THE HOUSE AT ANY LENGTH TODAY, AND I HOPE THAT ANY OF MY FRIENDS AND COLLEAGUES OR FORMER COLLEAGUES WHO ARE AFFECTED BY THE POLITICAL RECONSTRUCTION WILL MAKE ALL ALLOWANCES FOR ANY LACK OF CEREMONY WITH WHICH IT HAS BEEN NECESSARY TO ACT.
I SAY TO THE HOUSE AS I SAID TO MINISTERS WHO HAVE JOINED THIS GOVERNMENT, I HAVE NOTHING TO OFFER BUT BLOOD, TOIL, TEARS, AND SWEAT. WE HAVE BEFORE US AN ORDEAL OF THE MOST GRIEVOUS KIND. WE HAVE BEFORE US MANY, MANY MONTHS OF STRUGGLE AND SUFFERING.
YOU ASK, WHAT IS OUR POLICY? I SAY IT IS TO WAGE WAR BY LAND, SEA, AND AIR. WAR WITH ALL OUR MIGHT AND WITH ALL THE STRENGTH GOD HAS GIVEN US, AND TO WAGE WAR AGAINST A MONSTROUS TYRANNY NEVER SURPASSED IN THE DARK AND LAMENTABLE CATALOGUE OF HUMAN CRIME. THAT IS OUR POLICY.
YOU ASK, WHAT IS OUR AIM? I CAN ANSWER IN ONE WORD. IT IS VICTORY. VICTORY AT ALL COSTS – VICTORY IN SPITE OF ALL TERRORS – VICTORY, HOWEVER LONG AND HARD THE ROAD MAY BE, FOR WITHOUT VICTORY THERE IS NO SURVIVAL.
LET THAT BE REALIZED. NO SURVIVAL FOR THE BRITISH EMPIRE, NO SURVIVAL FOR ALL THAT THE BRITISH EMPIRE HAS STOOD FOR, NO SURVIVAL FOR THE URGE, THE IMPULSE OF THE AGES, THAT MANKIND SHALL MOVE FORWARD TOWARD HIS GOAL.
I TAKE UP MY TASK IN BUOYANCY AND HOPE. I FEEL SURE THAT OUR CAUSE WILL NOT BE SUFFERED TO FAIL AMONG MEN. I FEEL ENTITLED AT THIS JUNCTURE, AT THIS TIME, TO CLAIM THE AID OF ALL AND TO SAY, "COME THEN, LET US GO FORWARD TOGETHER WITH OUR UNITED STRENGTH."

Task 2b

Write an OpenCL program to perform the same functionality as in Task 2a, but in parallel. Note that it is more efficient to use OpenCL vector datatypes for processing in the kernel, as less work- items will be required.

```

__kernel void encryptKernel(__global char* plaintext, __global char* ciphertext, __global int* N){
    int i = get_global_id(0);
    if(plaintext[i] != ' '){
        if ((plaintext[i] < 'A') || (plaintext[i] > 'Z')){
            ciphertext[i] = plaintext[i];
        }
        else{
            ciphertext[i]=((plaintext[i]+N[i]-65)%26+'A');
            if(ciphertext[i] < 'A'){
                ciphertext[i] = 'Z' - ('A'-ciphertext[i]-1);
            }
        }
    }else{
        ciphertext[i] = ' ';
    }

}

__kernel void decryptKernel(__global char* ciphertext, __global char* plaintext, __global int* N){
    int i = get_global_id(0);
    if(ciphertext[i] != ' '){
        if ((ciphertext[i] < 'A') || (ciphertext[i] > 'Z')){
            plaintext[i] = ciphertext[i];
        }
        else{
            plaintext[i]=((ciphertext[i]-N[i]-65)%26+'A');
            if(plaintext[i] > 'Z')
                plaintext[i] = 'Z'- ('A'-plaintext[i]-1);

            if(plaintext[i] < 'A')
                plaintext[i] = 'Z' - ('A'-plaintext[i]-1);
        }
    }else{
        plaintext[i] = ' ';
    }
}

char reverseLookup(char in){
    switch (in){
        case 'G': return 'A';
        case 'X' : return 'B';
        case 'S': return 'C';
        case 'Q': return 'D';
        case 'F': return 'E';
        case 'A': return 'F';
        case 'R': return 'G';
        case 'O': return 'H';
        case 'W': return 'I';
        case 'B': return 'J';
        case 'L': return 'K';
        case 'M': return 'L';
        case 'T': return 'M';
        case 'H': return 'N';
        case 'C': return 'O';
        case 'V': return 'P';
        case 'P': return 'Q';
        case 'N': return 'R';
        case 'Z': return 'S';
        case 'U': return 'T';
        case 'I': return 'U';
        case 'E': return 'V';
        case 'Y': return 'W';
        case 'D': return 'X';
        case 'K': return 'Y';
        case 'J': return 'Z';
        case ' ': return ' ';
        default: return in;
    }
}
```

```
char lookupTable(char in){
    switch (in){
        case 'a':
        case 'A': return 'G';
        case 'b':
        case 'B': return 'X';
        case 'c':
        case 'C': return 'S';
        case 'd':
        case 'D': return 'Q';
        case 'e':
        case 'E': return 'F';
        case 'f':
        case 'F': return 'A';
        case 'g':
        case 'G': return 'R';
        case 'h':
        case 'H': return 'O';
        case 'i':
        case 'I': return 'W';
        case 'j':
        case 'J': return 'B';
        case 'k':
        case 'K': return 'L';
        case 'l':
        case 'L': return 'M';
        case 'm':
        case 'M': return 'T';
        case 'n':
        case 'N': return 'H';
        case 'o':
        case 'O': return 'C';
        case 'p':
        case 'P': return 'V';
        case 'q':
        case 'Q': return 'P';
        case 'r':
        case 'R': return 'N';
        case 's':
        case 'S': return 'Z';
        case 't':
        case 'T': return 'U';
        case 'u':
        case 'U': return 'I';
        case 'v':
        case 'V': return 'E';
        case 'w':
        case 'W': return 'Y';
        case 'x':
        case 'X': return 'D';
        case 'y':
        case 'Y': return 'K';
        case 'z':
        case 'Z': return 'J';
        case ' ': return ' ';
        default: return in;
    }
}
```

```
__kernel void KernelEncLookup(__global char* plaintext, __global char* ciphertext){
    int i = get_global_id(0);
    ciphertext[i] = lookupTable(plaintext[i]);
}

__kernel void KernelDecLookup(__global char* ciphertext, __global char* plaintext){
    int i = get_global_id(0);
    plaintext[i] = reverseLookup(ciphertext[i]);
}
```

//function to initialise vectors, create buffers, set work-units per kernel, enqueue kernel. Then displaying the vector created that is filled in parallel with the ***encrypted*** ciphertext, processed from the kernel that was programmed earlier on (as show in the above screenshot)

```
void encryptCL(std::vector<char> plaintext, cl::Program* program,
               cl::CommandQueue* queue, cl::Context* context,int shiftAmt){

    cl::Kernel kernel;
    cl::Event kernelEvent, readEvent;

    std::vector<cl_char>encryptVector;
    std::vector<cl_int> nVec(plaintext.size(),shiftAmt);


    kernel = cl::Kernel(*program, "encryptKernel");

    for (int i = 0; i < plaintext.size(); i++)
    {
        encryptVector.push_back(' ');
    }

    cl::Buffer plaintextBuffer = cl::Buffer(*context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
        sizeof(cl_char) * plaintext.size(), &plaintext[0]);
    cl::Buffer NBuffer = cl::Buffer(*context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(cl_int) *
        plaintext.size(), &nVec[0]);
    cl::Buffer writeBuffer = cl::Buffer(*context, CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(cl_char)
        * plaintext.size(), &encryptVector[0]);
    kernel.setArg(0,plaintextBuffer);
    kernel.setArg(1,writeBuffer);
    kernel.setArg(2,NBuffer);

    // enqueue
    cl::NDRange offset(0);
    cl::NDRange globalSize(plaintext.size());    // 4 work-units per kernel

    queue->enqueueNDRangeKernel(kernel, offset, globalSize);
    std::cout << "Kernel enqueued." << std::endl;

    std::cout << "--------------------------" << std::endl;
    std::cout << "Encrypting "<< std::endl;
    std::cout << "--------------------------" << std::endl;
    // enqueue command to read from device to host memory
    queue->enqueueReadBuffer(writeBuffer, CL_FALSE, 0, sizeof(cl_char) * plaintext.size(),
        &encryptVector[0], NULL, &readEvent);
    queue->finish();


    for(int i = 0; i < encryptVector.size(); i++){
        std::cout << encryptVector[i];
    }
    std::cout << std::endl;
```

//function to initialise vectors, create buffers, set work-units per kernel, enqueue kernel. Then displaying the vector created that is filled in parallel with the ***decrypted*** ciphertext, processed from the kernel that was programmed earlier on (as show in the above screenshot)

```
    //initialize a new vector for decrypted plaintext
    std::vector<cl_char>decryptedPlaintext;
     for (int i = 0; i < encryptVector.size(); i++)
    {
        decryptedPlaintext.push_back(' ');
    }

    cl::Buffer encryptedBuffer = cl::Buffer(*context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
        sizeof(cl_char) * encryptVector.size(), &encryptVector[0]);
    cl::Buffer decryptBuffer = cl::Buffer(*context, CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR,
        sizeof(cl_char) * decryptedPlaintext.size(), &decryptedPlaintext[0]);

    // create a kernel
    kernel = cl::Kernel(*program, "decryptKernel");
    kernel.setArg(0,encryptedBuffer);
    kernel.setArg(1,decryptBuffer);
    kernel.setArg(2,NBuffer);

    cl::NDRange globalSizeDec(encryptVector.size());    // 4 work-units per kernel

    queue->enqueueNDRangeKernel(kernel, offset, globalSizeDec);

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

    std::cout << "--------------------------" << std::endl;
    std::cout << "Decrypting "<< std::endl;
    std::cout << "--------------------------" << std::endl;     // enqueue command to read from device to
        host memory
    queue->enqueueReadBuffer(decryptBuffer, CL_FALSE, 0, sizeof(cl_char) * encryptVector.size(),
        &decryptedPlaintext[0], NULL, &readEvent);
    queue->finish();

    for(int i = 0; i < decryptedPlaintext.size(); i++){
        std::cout << decryptedPlaintext[i];
    }
    std::cout << std::endl;
}
```

Task 2c

Write an OpenCL program to perform parallel encryption, and decryption, by substituting characters based on the following lookup table:

Based on the table above, for encryption the letter a (or A) will be replaced by G, b (or B) will be replaced by X, c (or C) will be replaced by S, etc.

```
void encryptLookup(std::vector<char> loadedString, cl::Program* program,
        cl::CommandQueue* queue, cl::Context* context){
        cl::Kernel kernel;               // a single kernel object


        cl::Event kernelEvent, readEvent;

        std::vector<cl_char>encryptVector;

        kernel = cl::Kernel(*program, "kernelEncLookup");

        for (int i = 0; i < loadedString.size(); i++)
        {
            encryptVector.push_back('a');
        }

        cl::Buffer plaintextBuffer = cl::Buffer(*context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(cl_char) * loadedString.size(), &loadedString[0]);
        cl::Buffer writeBuffer = cl::Buffer(*context, CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(cl_char) * loadedString.size(), &encryptVector[0]);
        kernel.setArg(0,plaintextBuffer);
        kernel.setArg(1,writeBuffer);

        // enqueue
        cl::NDRange offset(0);
        cl::NDRange globalSize(loadedString.size());    // 4 work-units per kernel

        queue->enqueueNDRangeKernel(kernel, offset, globalSize);


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

        std::cout << "Decrypting "<< std::endl;
        std::cout << "--------------------------" << std::endl;
    // enqueue command to read from device to host memory
        queue->enqueueReadBuffer(writeBuffer, CL_FALSE, 0, sizeof(cl_char) * loadedString.size(), &encryptVector[0], NULL, &readEvent);
        queue->finish();


        for(int i = 0; i < encryptVector.size(); i++){
            std::cout << encryptVector[i];
        }
        std::cout << std::endl;



        std::vector<cl_char>decryptedPlaintext;
         for (int i = 0; i < encryptVector.size(); i++)
        {
            decryptedPlaintext.push_back('a');
        }

        cl::Buffer encryptedBuffer = cl::Buffer(*context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(cl_char) * encryptVector.size(), &encryptVector[0]);
        cl::Buffer decryptBuffer = cl::Buffer(*context, CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(cl_char) * decryptedPlaintext.size(), &decryptedPlaintext[0]);


       kernel = cl::Kernel(*program, "KernelDecLookup");
       kernel.setArg(0,encryptedBuffer);
       kernel.setArg(1,decryptBuffer);

       cl::NDRange globalSizeDec(encryptVector.size());    // 4 work-units per kernel

       queue->enqueueNDRangeKernel(kernel, offset, globalSizeDec);



       std::cout << "Kernel enqueued." << std::endl;
       std::cout << "--------------------------" << std::endl;
      std::cout << "Decrypting "<< std::endl;
      std::cout << "--------------------------" << std::endl;
        // enqueue command to read from device to host memory
       queue->enqueueReadBuffer(decryptBuffer, CL_FALSE, 0, sizeof(cl_char) * encryptVector.size(), &decryptedPlaintext[0], NULL, &readEvent);
       queue->finish();

       for(int i = 0; i < decryptedPlaintext.size(); i++){
           std::cout << decryptedPlaintext[i];
       }
       std::cout << std::endl;
}
```

//function to initialise vectors, create buffers, set work-units per kernel, enqueue kernel. Then displaying the vector created that is filled in parallel with the ***decrypted*** ciphertext using the lookup table set up in the kernel (where by the condition is set for each of the ciphertext found, a corresponding text will be returned. (Lookup table provided from the assignment specifications)).

// Kernel codes for lookup table below:

```
char reverseLookup(char in){
    switch (in){
        case 'G': return 'A';
        case 'X' : return 'B';
        case 'S': return 'C';
        case 'Q': return 'D';
        case 'F': return 'E';
        case 'A': return 'F';
        case 'R': return 'G';
        case 'O': return 'H';
        case 'W': return 'I';
        case 'B': return 'J';
        case 'L': return 'K';
        case 'M': return 'L';
        case 'T': return 'M';
        case 'H': return 'N';
        case 'C': return 'O';
        case 'V': return 'P';
        case 'P': return 'Q';
        case 'N': return 'R';
        case 'Z': return 'S';
        case 'U': return 'T';
        case 'I': return 'U';
        case 'E': return 'V';
        case 'Y': return 'W';
        case 'D': return 'X';
        case 'K': return 'Y';
        case 'J': return 'Z';
        case ' ': return ' ';
        default: return in;
    }
}
```

```
char lookupTable(char in){
    switch (in){
        case 'a':
        case 'A': return 'G';
        case 'b':
        case 'B': return 'X';
        case 'c':
        case 'C': return 'S';
        case 'd':
        case 'D': return 'Q';
        case 'e':
        case 'E': return 'F';
        case 'f':
        case 'F': return 'A';
        case 'g':
        case 'G': return 'R';
        case 'h':
        case 'H': return 'O';
        case 'i':
        case 'I': return 'W';
        case 'j':
        case 'J': return 'B';
        case 'k':
        case 'K': return 'L';
        case 'l':
        case 'L': return 'M';
        case 'm':
        case 'M': return 'T';
        case 'n':
        case 'N': return 'H';
        case 'o':
        case 'O': return 'C';
        case 'p':
        case 'P': return 'V';
        case 'q':
        case 'Q': return 'P';
        case 'r':
        case 'R': return 'N';
        case 's':
        case 'S': return 'Z';
        case 't':
        case 'T': return 'U';
        case 'u':
        case 'U': return 'I';
        case 'v':
        case 'V': return 'E';
        case 'w':
        case 'W': return 'Y';
        case 'x':
        case 'X': return 'D';
        case 'y':
        case 'Y': return 'K';
        case 'z':
        case 'Z': return 'J';
        case ' ': return ' ';
        default: return in;
    }
}
```

```
__kernel void KernelEncLookup(__global char* plaintext, __global char* ciphertext){
    int i = get_global_id(0);
    ciphertext[i] = lookupTable(plaintext[i]);
}

__kernel void KernelDecLookup(__global char* ciphertext, __global char* plaintext){
    int i = get_global_id(0);
    plaintext[i] = reverseLookup(ciphertext[i]);
}
```

//Normal Encryption & Decryption

```
-- Loading plaintext.txt --
plaintext.txt loaded
Name: Yeo Zheng Xu Isaac
Student ID: 6342425
------------------------------------
     1. -- 2a. Run using C++ program
     2. -- 2b. Run using OpenCL program
     3. -- 2c. Run OpenCL Lookup table program
Select an option: 1
Select a shift amount: 5

Encrypting into output.txt
Done

-- Decrypting into decrypted.txt --
Done
Program ended with exit code: 0
```

//Encryption & Decryption using OpenCL

```
-- Loading plaintext.txt --
plaintext.txt loaded
Name: Yeo Zheng Xu Isaac
Student ID: 6342425
---------------------------------------
     1. -- 2a. Run using C++ program
     2. -- 2b. Run using OpenCL program
     3. -- 2c. Run OpenCL Lookup table program
Select an option: 2
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: 1

Creating context...
Context created

Creating command queue...
Command queue created
Program build: Successful
--------------------
Select a shit amount: 5
Kernel enqueued.
--------------------------
Encrypting
--------------------------
GQTTI, XBJFY FSI YJFWX XUJJHM GD XNW BNSXYTS HMZWHMNQQ

N YWZXY BMJS UFWQNFRJSY RJJYX FLFNS YMNX UFWY TK RD YFXP BNQQ GJ HTRUQJYJI FSI YMFY
YMJ FIRNSNXYWFYNTS BNQQ GJ HTRUQJYJ NS FQQ WJXUJHYX. N HTSXNIJWJI NY NS YMJ UZGQNH
NSYJWJXY YT XZLLJXY YT YMJ XUJFPJW YMFY YMJ MTZXJ XMTZQI GJ XZRRTSJI YTIFD. FY YMJ
JSI TK YTIFD'X UWTHJJINSLX, YMJ FIOTZWSRJSY TK YMJ MTZXJ BNQQ GJ UWTUTXJI ZSYNQ RFD
21 BNYM UWTANXNTS KTW JFWQNJW RJJYNSL NK SJJI GJ. GZXNSJXX KTW YMFY BNQQ GJ STYNKNJI
YT RUX FY YMJ JFWQNJXY TUUTWYZSNYD.

N STB NSANYJ YMJ MTZXJ GD F WJXTQZYNTS YT WJHTWI NYX FUUWTAFQ TK YMJ XYJUX YFPJS FSI
IJHQFWJ NYX HTSKNIJSHJ NS YMJ SJB LTAJWSRJSY.

YMJ WJXTQZYNTS:
"YMFY YMNX MTZXJ BJQHTRJX YMJ KTWRFYNTS TK F LTAJWSRJSY WJUWJXJSYNSL YMJ ZSNYJI FSI
NSKQJCNGQJ WJXTQAJ TK YMJ SFYNTS YT UWTXJHZYJ YMJ BFW BNYM LJWRFSD YT F ANHYTWNTZX
HTSHQZXNTS."

YT KTWR FS FIRNSNXYWFYNTS TK YMNX XHFQJ FSI HTRUQJCNYD NX F XJWNTZX ZSIJWYFPNSL NS
NYXJQK. GZY BJ FWJ NS YMJ UWJQNRNSFWD UMFXJ TK TSJ TK YMJ LWJFYJXY GFYYQJX NS MNXYTWD.
BJ FWJ NS FHYNTS FY RFSD TYMJW UTNSYX-NS STWBFD FSI NS MTQQFSI-FSI BJ MFAJ YT GJ
UWJUFWJI NS YMJ RJINYJWWFSJFS. YMJ FNW GFYYQJ NX HTSYNSZNSL, FSI RFSD UWJUFWFYNTSX
MFAJ YT GJ RFIJ MJWJ FY MTRJ.

NS YMNX HWNXNX N YMNSP N RFD GJ UFWITSJI NK 1 IT STY FIIWJXX YMJ MTZXJ FY FSD QJSLYM
YTIFD, FSI N MTUJ YMFY FSD TK RD KWNJSIX FSI HTQQJFLZJX TW KTWRJW HTQQJFLZJX BMT FWJ
FKKJHYJI GD YMJ UTQNYNHFQ WJHTSXYWZHYNTS BNQQ RFPJ FQQ FQQTBFSHJX KTW FSD QFHP TK
HJWJRTSD BNYM BMNHM NY MFX GJJS SJHJXXFWD YT FHY.

N YFP UT UMJ MTZXJ FY N YFNT UT BNSNUUQJU BMT MFAJ QTNSJI YMNU LTAJWSRJSY. N MFAJ
DTZ FXP, BMFY NX TZW UTQNHD? N XFD NY NX YT BFLJ BFW GD QFSI, XJF, FSI FNW. BFW BNYM
FQQ TZW RNLMY FSI BNYM FQQ YMJ XYWJSLYM LTI MFX LNAJS ZX, FSI YT BFLJ BFW FLFNSXY F
RTSXYWTZX YDWFSSD SJAJW XZWUFXXJI NS YMJ IFWP FSI QFRJSYFGQJ HFYFQTLZJ TK MZRFS HWNRJ.
YMFY NX TZW UTQNHD.

DTZ FXP, BMFY NX TZW FNR? N HFS FSXBJW NS TSJ BTWI. NY NX ANHYTWD. ANHYTWD FY FQQ HTXYX
- ANHYTWD NS XUNYJ TK FQQ YJWWTWX - ANHYTWD, MTBJAJW QTSL FSI MFWI YMJ WTFI RFD GJ, KTW
BNYMTZY ANHYTWD YMJWJ NX ST XZWANAFQ.

QJY YMFY GJ WJFQNEJI. ST XZWANAFQ KTW YMJ GWNYNXM JRUNWJ, ST XZWANAFQ KTW FQQ YMFY YMJ
GWNYNXM JRUNWJ MFX XYTTI KTW, ST XZWANAFQ KTW YMJ ZWLJ, YMJ NRUZQXJ TK YMJ FLJX, YMFY
RFSPNSI XMFQQ RTAJ KTWBFWI YTBFWI MNX LTFQ.

N YFPJ ZU RD YFXP NS GZTDFSHD FSI MTUJ. N KJJQ XZWJ YMFY TZW HFZXJ BNQQ STY GJ XZKKJWJI
YT KFNQ FRTSL RJS. N KJJQ JSYNYQJI FY YMNX OZSHYZWJ, FY YMNX YNRJ, YT HQFNR YMJ FNI TK
FQQ FSI YT XFD, "HTRJ YMJS, QJY ZX LT KTWBFWI YTLJYMJW BNYM TZW ZSNYJI XYWJSLYM."
```

```
Kernel enqueued.
--------------------------
Decrypting
--------------------------
BLOOD, SWEAT AND TEARS SPEECH BY SIR WINSTON CHURCHILL

I TRUST WHEN PARLIAMENT MEETS AGAIN THIS PART OF MY TASK WILL BE COMPLETED AND THAT
THE ADMINISTRATION WILL BE COMPLETE IN ALL RESPECTS. I CONSIDERED IT IN THE PUBLIC
INTEREST TO SUGGEST TO THE SPEAKER THAT THE HOUSE SHOULD BE SUMMONED TODAY. AT THE
END OF TODAY'S PROCEEDINGS, THE ADJOURNMENT OF THE HOUSE WILL BE PROPOSED UNTIL MAY
21 WITH PROVISION FOR EARLIER MEETING IF NEED BE. BUSINESS FOR THAT WILL BE NOTIFIED
TO MPS AT THE EARLIEST OPPORTUNITY.

I NOW INVITE THE HOUSE BY A RESOLUTION TO RECORD ITS APPROVAL OF THE STEPS TAKEN AND
DECLARE ITS CONFIDENCE IN THE NEW GOVERNMENT.

THE RESOLUTION:
"THAT THIS HOUSE WELCOMES THE FORMATION OF A GOVERNMENT REPRESENTING THE UNITED AND
INFLEXIBLE RESOLVE OF THE NATION TO PROSECUTE THE WAR WITH GERMANY TO A VICTORIOUS
CONCLUSION."

TO FORM AN ADMINISTRATION OF THIS SCALE AND COMPLEXITY IS A SERIOUS UNDERTAKING IN
ITSELF. BUT WE ARE IN THE PRELIMINARY PHASE OF ONE OF THE GREATEST BATTLES IN HISTORY.
WE ARE IN ACTION AT MANY OTHER POINTS—IN NORWAY AND IN HOLLAND—AND WE HAVE TO BE
PREPARED IN THE MEDITERRANEAN. THE AIR BATTLE IS CONTINUING, AND MANY PREPARATIONS
HAVE TO BE MADE HERE AT HOME.

IN THIS CRISIS I THINK I MAY BE PARDONED IF 1 DO NOT ADDRESS THE HOUSE AT ANY LENGTH
TODAY, AND I HOPE THAT ANY OF MY FRIENDS AND COLLEAGUES OR FORMER COLLEAGUES WHO ARE
AFFECTED BY THE POLITICAL RECONSTRUCTION WILL MAKE ALL ALLOWANCES FOR ANY LACK OF
CEREMONY WITH WHICH IT HAS BEEN NECESSARY TO ACT.

I SAY TO THE HOUSE AS I SAID TO MINISTERS WHO HAVE JOINED THIS GOVERNMENT, I HAVE
NOTHING TO OFFER BUT BLOOD, TOIL, TEARS, AND SWEAT. WE HAVE BEFORE US AN ORDEAL OF
THE MOST GRIEVOUS KIND. WE HAVE BEFORE US MANY, MANY MONTHS OF STRUGGLE AND SUFFERING.

YOU ASK, WHAT IS OUR POLICY? I SAY IT IS TO WAGE WAR BY LAND, SEA, AND AIR. WAR WITH
ALL OUR MIGHT AND WITH ALL THE STRENGTH GOD HAS GIVEN US, AND TO WAGE WAR AGAINST A
MONSTROUS TYRANNY NEVER SURPASSED IN THE DARK AND LAMENTABLE CATALOGUE OF HUMAN CRIME.
THAT IS OUR POLICY.

YOU ASK, WHAT IS OUR AIM? I CAN ANSWER IN ONE WORD. IT IS VICTORY. VICTORY AT ALL COSTS
– VICTORY IN SPITE OF ALL TERRORS – VICTORY, HOWEVER LONG AND HARD THE ROAD MAY BE, FOR
WITHOUT VICTORY THERE IS NO SURVIVAL.

LET THAT BE REALIZED. NO SURVIVAL FOR THE BRITISH EMPIRE, NO SURVIVAL FOR ALL THAT THE
BRITISH EMPIRE HAS STOOD FOR, NO SURVIVAL FOR THE URGE, THE IMPULSE OF THE AGES, THAT
MANKIND SHALL MOVE FORWARD TOWARD HIS GOAL.

I TAKE UP MY TASK IN BUOYANCY AND HOPE. I FEEL SURE THAT OUR CAUSE WILL NOT BE SUFFERED
TO FAIL AMONG MEN. I FEEL ENTITLED AT THIS JUNCTURE, AT THIS TIME, TO CLAIM THE AID OF
ALL AND TO SAY, "COME THEN, LET US GO FORWARD TOGETHER WITH OUR UNITED STRENGTH."
```

//Custom Encryption & Decryption using openCL with lookup table specified

```
-- Loading plaintext.txt --
plaintext.txt loaded
Name: Yeo Zheng Xu Isaac
Student ID: 6342425
------------------------------------
     1. -- 2a. Run using C++ program
     2. -- 2b. Run using OpenCL program
     3. -- 2c. Run OpenCL Lookup table program
Select an option: 3
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: 1

Creating context...
Context created

Creating command queue...
Command queue created
Program build: Successful
--------------------
Kernel enqueued.
--------------------------
Decrypting
--------------------------
XMCCQ, ZYFGU GHQ UFGNZ ZVFFSO XK ZWN YWHZUCH SOINSOWMM

W UNIZU YOFH VGNMWGTFHU TFFUZ GRGWH UOWZ VGNU CA TK UGZL YWMM XF SCTVMFUFQ GHQ UOGU
UOF GQTWHWZUNGUWCH YWMM XF SCTVMFUF WH GMM NFZVFSUZ. W SCHZWQFNFQ WU WH UOF VIXMWS
WHUFNFZU UC ZIRRFZU UC UOF ZVFGLFN UOGU UOF OCIZF ZOCIMQ XF ZITTCHFQ UCQGK. GU UOF
FHQ CA UCQGK'Z VNCSFFQWHRZ, UOF GQBCINHTFHU CA UOF OCIZF YWMM XF VNCVCZFQ IHUWM TGK
21 YWUO VNCEWZWCH ACN FGNMWFN TFFUWHR WA HFFQ XF. XIZWHFZZ ACN UOGU YWMM XF HCUWAWFQ
UC TVZ GU UOF FGNMWFZU CVVCNUIHWUK.

W HCY WHEWUF UOF OCIZF XK G NFZCMIUWCH UC NFSCNQ WUZ GVVNCEGM CA UOF ZUFVZ UGLFH GHQ
QFSMGNF WUZ SCHAWQFHSF WH UOF HFY RCEFNHTFHU.

UOF NFZCMIUWCH:
"UOGU UOWZ OCIZF YFMSCTFZ UOF ACNTGUWCH CA G RCEFNHTFHU NFVNFZFHUWHR UOF IHWUFQ GHQ
WHAMFDWXMF NFZCMEF CA UOF HGUWCH UC VNCZFSIUF UOF YGN YWUO RFNTGHK UC G EWSUCNWCIZ
SCHSMIZWCH."

UC ACNT GH GQTWHWZUNGUWCH CA UOWZ ZSGMF GHQ SCTVMFDWUK WZ G ZFNWCIZ IHQFNUGLWHR WH
WUZFMA. XIU YF GNF WH UOF VNFMWTWHGNK VOGZF CA CHF CA UOF RNFGUFZU XGUUMFZ WH OWZUCNK.
YF GNF WH GSUWCH GU TGHK CUOFN VCWHUZ-WH HCNYGK GHQ WH OCMMGHQ-GHQ YF OGEF UC XF
VNFVGNFQ WH UOF TFQWUFNNGHFGH. UOF GWN XGUUMF WZ SCHUWHIWHR, GHQ TGHK VNFVGNGUWCHZ
OGEF UC XF TGQF OFNF GU OCTF.

WH UOWZ SNWZWZ W UOWHL W TGK XF VGNQCHFQ WA 1 QC HCU GQQNFZZ UOF OCIZF GU GHK MFHRUO
UCQGK, GHQ W OCVF UOGU GHK CA TK ANWFHQZ GHQ SCMMFGRIFZ CN ACNTFN SCMMFGRIFZ YOC GNF
GAAFSUFQ XK UOF VCMWUWSGM NFSCHZUNISUWCH YWMM TGLF GMM GMMCYGHSFZ ACN GHK MGSL CA
SFNFTCHK YWUO YOWSO WU OGZ XFFH HFSFZZGNK UC GSU.

W ZGK UC UOF OCIZF GZ W ZGWQ UC TWHWZUFNZ YOC OGEF BCWHFQ UOWZ RCEFNHTFHU, W OGEF
HCUOWHR UC CAAFN XIU XMCCQ, UCWM, UFGNZ, GHQ ZYFGU. YF OGEF XFACNF IZ GH CNQFGM CA
UOF TCZU RNWFECIZ LWHQ. YF OGEF XFACNF IZ TGHK, TGHK TCHUOZ CA ZUNIRRMF GHQ ZIAAFNWHR.

KCI GZL, YOGU WZ CIN VCMWSK? W ZGK WU WZ UC YGRF YGN XK MGHQ, ZFG, GHQ GWN. YGN YWUO
GMM CIN TWROU GHQ YWUO GMM UOF ZUNFHRUO RCQ OGZ RWEFH IZ, GHQ UC YGRF YGN GRGWHZU G
TCHZUNCIZ UKNGHHK HFEFN ZINVGZZFQ WH UOF QGNL GHQ MGTFHUGXMF SGUGMCRIF CA OITGH SNWTF.
UOGU WZ CIN VCMWSK.

KCI GZL, YOGU WZ CIN GWT? W SGH GHZYFN WH CHF YCNQ. WU WZ EWSUCNK. EWSUCNK GU GMM SCZUZ
- EWSUCNK WH ZVWUF CA GMM UFNNCNZ - EWSUCNK, OCYFEFN MCHR GHQ OGNQ UOF NCGQ TGK XF, ACN
YWUOCIU EWSUCNK UOFNF WZ HC ZINEWEGM.

MFU UOGU XF NFGMWJFQ. HC ZINEWEGM ACN UOF XNWUWZO FTVWNF, HC ZINEWEGM ACN GMM UOGU UOF
XNWUWZO FTVWNF OGZ ZUCCQ ACN, HC ZINEWEGM ACN UOF INRF, UOF WTVIMZF CA UOF GRFZ, UOGU
TGHLWHQ ZOGMM TCEF ACNYGNQ UCYGNQ OWZ RCGM.

W UGLF IV TK UGZL WH XICKGHSK GHQ OCVF. W AFFM ZINF UOGU CIN SGIZF YWMM HCU XF ZIAAFNFQ
UC AGWM GTCHR TFH. W AFFM FHUWUMFQ GU UOWZ BIHSUINF, GU UOWZ UWTF, UC SMGWT UOF GWQ CA
GMM GHQ UC ZGK, "SCTF UOFH, MFU IZ RC ACNYGNQ UCRFUOFN YWUO CIN IHWUFQ ZUNFHRUO."
```

```
Kernel enqueued.
---------------------------
Decrypting
---------------------------
BLOOD, SWEAT AND TEARS SPEECH BY SIR WINSTON CHURCHILL

I TRUST WHEN PARLIAMENT MEETS AGAIN THIS PART OF MY TASK WILL BE COMPLETED AND THAT
THE ADMINISTRATION WILL BE COMPLETE IN ALL RESPECTS. I CONSIDERED IT IN THE PUBLIC
INTEREST TO SUGGEST TO THE SPEAKER THAT THE HOUSE SHOULD BE SUMMONED TODAY. AT THE
END OF TODAY'S PROCEEDINGS, THE ADJOURNMENT OF THE HOUSE WILL BE PROPOSED UNTIL MAY
21 WITH PROVISION FOR EARLIER MEETING IF NEED BE. BUSINESS FOR THAT WILL BE NOTIFIED
TO MPS AT THE EARLIEST OPPORTUNITY.

I NOW INVITE THE HOUSE BY A RESOLUTION TO RECORD ITS APPROVAL OF THE STEPS TAKEN AND
DECLARE ITS CONFIDENCE IN THE NEW GOVERNMENT.

THE RESOLUTION:
"THAT THIS HOUSE WELCOMES THE FORMATION OF A GOVERNMENT REPRESENTING THE UNITED AND
INFLEXIBLE RESOLVE OF THE NATION TO PROSECUTE THE WAR WITH GERMANY TO A VICTORIOUS
CONCLUSION."

TO FORM AN ADMINISTRATION OF THIS SCALE AND COMPLEXITY IS A SERIOUS UNDERTAKING IN
ITSELF. BUT WE ARE IN THE PRELIMINARY PHASE OF ONE OF THE GREATEST BATTLES IN HISTORY.
WE ARE IN ACTION AT MANY OTHER POINTS—IN NORWAY AND IN HOLLAND—AND WE HAVE TO BE
PREPARED IN THE MEDITERRANEAN. THE AIR BATTLE IS CONTINUING, AND MANY PREPARATIONS
HAVE TO BE MADE HERE AT HOME.

IN THIS CRISIS I THINK I MAY BE PARDONED IF 1 DO NOT ADDRESS THE HOUSE AT ANY LENGTH
TODAY, AND I HOPE THAT ANY OF MY FRIENDS AND COLLEAGUES OR FORMER COLLEAGUES WHO ARE
AFFECTED BY THE POLITICAL RECONSTRUCTION WILL MAKE ALL ALLOWANCES FOR ANY LACK OF
CEREMONY WITH WHICH IT HAS BEEN NECESSARY TO ACT.

I SAY TO THE HOUSE AS I SAID TO MINISTERS WHO HAVE JOINED THIS GOVERNMENT, I HAVE
NOTHING TO OFFER BUT BLOOD, TOIL, TEARS, AND SWEAT. WE HAVE BEFORE US AN ORDEAL OF
THE MOST GRIEVOUS KIND. WE HAVE BEFORE US MANY, MANY MONTHS OF STRUGGLE AND SUFFERING.

YOU ASK, WHAT IS OUR POLICY? I SAY IT IS TO WAGE WAR BY LAND, SEA, AND AIR. WAR WITH
ALL OUR MIGHT AND WITH ALL THE STRENGTH GOD HAS GIVEN US, AND TO WAGE WAR AGAINST A
MONSTROUS TYRANNY NEVER SURPASSED IN THE DARK AND LAMENTABLE CATALOGUE OF HUMAN CRIME.
THAT IS OUR POLICY.

YOU ASK, WHAT IS OUR AIM? I CAN ANSWER IN ONE WORD. IT IS VICTORY. VICTORY AT ALL COSTS
– VICTORY IN SPITE OF ALL TERRORS – VICTORY, HOWEVER LONG AND HARD THE ROAD MAY BE, FOR
WITHOUT VICTORY THERE IS NO SURVIVAL.

LET THAT BE REALIZED. NO SURVIVAL FOR THE BRITISH EMPIRE, NO SURVIVAL FOR ALL THAT THE
BRITISH EMPIRE HAS STOOD FOR, NO SURVIVAL FOR THE URGE, THE IMPULSE OF THE AGES, THAT
MANKIND SHALL MOVE FORWARD TOWARD HIS GOAL.

I TAKE UP MY TASK IN BUOYANCY AND HOPE. I FEEL SURE THAT OUR CAUSE WILL NOT BE SUFFERED
TO FAIL AMONG MEN. I FEEL ENTITLED AT THIS JUNCTURE, AT THIS TIME, TO CLAIM THE AID OF
ALL AND TO SAY, "COME THEN, LET US GO FORWARD TOGETHER WITH OUR UNITED STRENGTH."
Program ended with exit code: 0
```

Task 3 – Parallel Image Processing

Task 3a (Image Luminance)

Write a parallel program to convert the RGB values (i.e. Red, Green and Blue colour channels) in an image to luminance values (this approach is used to convert a colour image into a greyscale image).

For each pixel, calculate:
Luminance = 0.299*R + 0.587*G + 0.114*B

Save the luminance image into a 24-bit BMP file. To do this, set the RGB values of each pixel to the luminance value.

Note that if you use the example code from the tutorial on image processing, the R, G, and B values range from 0 to 255 on the host (unsigned char), and 0.0 to 1.0 (float) on the device.

.cpp file (read in peppers.bmp file, create buffers, command enqueue)

```
    try
    {

        if (!select_one_device(&platform, &devices))
        {
            // if no device selected
            quit_program("Device not selected.");
        }
        else
        {
            //create details
            details = Context(devices);
            cout << endl;
            cout << "Checking if programming is built successfully..." << endl;
            bool programBuilt = build_program(&programming, &details, "/Users/isaacyeo/Desktop/A2Task3/A2Task3/Task3a.cl");
            if (programBuilt == false)
            {
                // if OpenCL program build error
                quit_program("OpenCL program build error.");
            }
            else
            {
//
                string fileName = "/Users/isaacyeo/Desktop/A2Task3/A2Task3/peppers.bmp";
//

                vector <float> luminanceVector;

                //convert the RGB values in an image to luminance values
                //set each RGB to luminance values and save image
                BMP_RGB_to_Luminous(fileName, &width, &height, luminanceVector);


                //find total number of pixels
                int numOfElements = width * height;
                //store all information to vector of cl_float
                vector <cl_float>imageData(numOfElements);
                //transfer all information from luminaceVector to imageData
                for (unsigned int i = 0; i < luminanceVector.size(); i++)
                {
                    imageData[i] = luminanceVector[i];
                }

                //create kernel
                kernel = Kernel(programming, "reduction");
                //create command queue
                queue = CommandQueue(details, devices);
                //get information
                workGroupSize = devices.getInfo<CL_DEVICE_MAX_WORK_GROUP_SIZE>();
                localMemSize = devices.getInfo<CL_DEVICE_LOCAL_MEM_SIZE>();
                kWorkGroupSize = kernel.getWorkGroupInfo<CL_KERNEL_WORK_GROUP_SIZE>(devices);


            }
        }
    }
```

//bmpfuncs.cpp

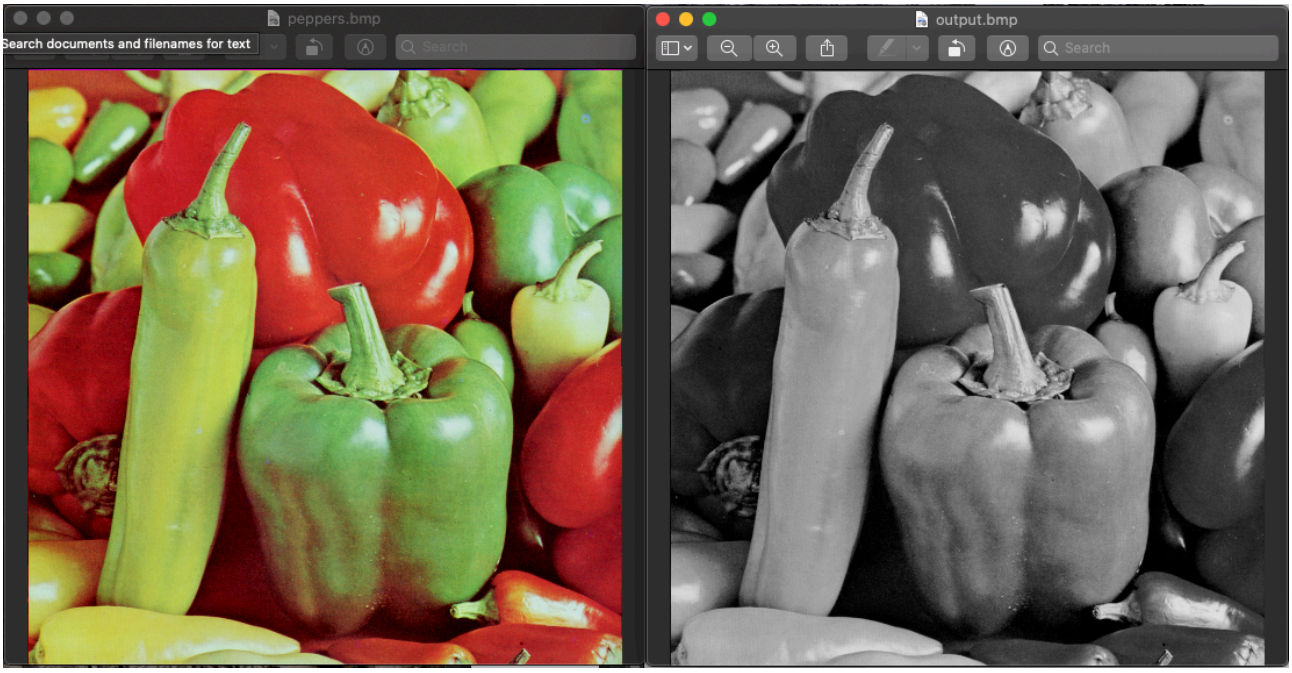
Covert image's RGB values to luminance values as shown below:

Luminance = 0.299*R + 0.587*G + 0.114*B

—> Output greyscale image to file path

```cpp
#include "bmpfuncs.h"

// reads the contents of a 24-bit RGB bitmap file and returns it in RGBA format
void BMP_RGB_to_Luminous(string filename, int* widthOut, int* heightOut, vector<float> &luminance)
{
    char fileHeader[54];
    int width, height;
    int offset, imageSize, fileSize;
    char colour[3];
    unsigned char* imageData;

    // open image file
    ifstream getImageFile(filename, ios::in | ios::binary);
    // check whether file opened successfully
    if (getImageFile.is_open() == false)
    {
        cout << "Failed to open texture file - " << filename << endl;
    }
    //process file header
    getImageFile.read(fileHeader, 54);

    //get information for offset, width and height
    offset = fileHeader[10];
    width = fileHeader[21] << 24;
    width |= fileHeader[20] << 16;
    width |= fileHeader[19] << 8;
    width |= fileHeader[18];
    width = abs(width);
    height = fileHeader[25] << 24;
    height |= fileHeader[24] << 16;
    height |= fileHeader[23] << 8;
    height |= fileHeader[22];
    height = abs(height);

    //create imageData to store RGB values of each pixels
    imageData = new unsigned char[width*height * 3];
    //get each rgb value
    int rgbRow = width * 3;
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < rgbRow; j += 3) {
            getImageFile.read(colour, 3);
            imageData[i*rgbRow + j] = colour[0];
            imageData[i*rgbRow + j + 1] = colour[1];
            imageData[i*rgbRow + j + 2] = colour[2];
        }
    }
    //close file
    getImageFile.close();

    //create file to be exported
    ofstream outFileStream("/Users/isaacyeo/Desktop/CSCI376_A2_6342425_YeoZhengXuIsaac/A2TASK3/A2Task3/A2Task3/output.bmp", ios::out |
        ios::binary);
    if (outFileStream.is_open() == false)
    {
        cout << "Failed to open output file - " << filename << endl;
    }
```

```
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: 1

Checking if programming is built successfully...
Program build: Successful
--------------------
Output File Generated in Assignment Folder - /Users/isaacyeo/Desktop/A2Task3/A2Task3/peppers.bmp
```



Task 3b (Gaussian Blurring)

Gaussian blurring is a commonly used technique to image processing and graphics to create a smooth blurring effect using a Gaussian function. The weights of the filter depend on the size of the

---

Gaussian filter window. The following are example weights

for a 7x7 windows:

0.000036 0.000363 0.001446 0.002291 0.001446 0.000363 0.000036

0.000363 0.003676 0.014662 0.023226 0.014662 0.003676 0.000363

0.001446 0.014662 0.058488 0.092651 0.058488 0.014662 0.001446

0.002291 0.023226 0.092651 0.146768 0.092651 0.023226 0.002291

0.001446 0.014662 0.058488 0.092651 0.058488 0.014662 0.001446

0.000363 0.003676 0.014662 0.023226 0.014662 0.003676 0.000363

0.000036 0.000363 0.001446 0.002291 0.001446 0.000363 0.000036

i. Write an OpenCL program that
Gaussian blurring based on the 7x7 window weights provided above.

accepts a colour image and outputs a filtered image using

ii. Instead of using the 7x7 window (the naïve approach), an alternate approach is to run the filter in 2 passes. The first pass will perform blurring in the horizontal direction; the result will then undergo a second pass to blur it in the vertical direction (enqueue the kernel twice to perform blurring in each direction). The result will be similar to the single window approach, but the amount of computation will be different.

For example, using a 7x7 window approach, each pixel will have to perform a weighted sum on 49 pixels. In the 2-pass approach, each pixel will have to perform a weighted sum on 7 pixels in each pass, processing a total of 14 pixels. This is illustrated below:

//kernel with weights for both the Blurring Filter 7x7 & 2-Pass Filter

```
__constant sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;


__constant float BlurringFilter7x7[49] = { 0.000036, 0.000363, 0.001446, 0.002291, 0.001446, 0.000363, 0.000036,
                                           0.000363, 0.003676, 0.014662, 0.023226, 0.014662, 0.003676, 0.000363,
                                           0.001446, 0.014662, 0.058488, 0.092651, 0.058488, 0.014662, 0.001446,
                                           0.002291, 0.023226, 0.092651, 0.146768, 0.092651, 0.023226, 0.002291,
                                           0.001446, 0.014662, 0.058488, 0.092651, 0.058488, 0.014662, 0.001446,
                                           0.000363, 0.003676, 0.014662, 0.023226, 0.014662, 0.003676, 0.000363,
                                           0.000036, 0.000363, 0.001446, 0.002291, 0.001446, 0.000363, 0.000036 };


__constant float GaussianBlur2PassFilter7x7[7] = { 0.00598, 0.060626, 0.241843, 0.383103, 0.241843, 0.060626, 0.00598 };

__kernel void blur_image(__global int* choices, read_only image2d_t src_image,
                         write_only image2d_t dst_image)
{
    //get option chosen by user
    int UserChoices = choices[0];
    int min = 0;
    int max = 0;

    // Accumulated pixel value
    float4 sum = (float4)(0.0);

    //Set filter to 0
    int index = 0;

    int2 coord;
    float4 pixel;

    //if user chose option window pass approach

    //set range value in for loop based on choices by user

        min = -3;
        max = 3;
```

```
        //Iterate over the rows
    if (UserChoices < 4)
    {
        //Get row and column position
        int column = get_global_id(0);
        int row = get_global_id(1);

        for (int i = min; i <= max; i++) {
            coord.y = row + i;

            //Iterate over the columns
            for (int j = min; j <= max; j++) {
                coord.x = column + j;

                //Read pixel from the image
                pixel = read_imagef(src_image, sampler, coord);
                //Acculumate weighted sum

                    sum.xyz += pixel.xyz * BlurringFilter7x7[index++];


            }
        }
        //Write pixel to output
        coord = (int2)(column, row);
        write_imagef(dst_image, coord, sum);

    }
    //if user chose option window pass approach
    else

    {

        int column = get_global_id(0);
        int row = get_global_id(1);
        coord.y = row;
        index = 0;
        //Iterate over the columns
        for (int j = min; j <= max; j++)
        {
            coord.x = column + j;

            //Read pixel from the image
            pixel = read_imagef(src_image, sampler, coord);
            //Acculumate weighted sum

                sum.xyz += pixel.xyz * GaussianBlur2PassFilter7x7[index++];

        }
        coord.x = column;
        index = 0;
        //Iterate over the columns
        //Iterate over the columns
        for (int j = min; j <= max; j++)
        {
            coord.y = row + j;

            //Read pixel from the image
            pixel = read_imagef(src_image, sampler, coord);

        }
        //Write pixel to output
        coord = (int2)(column, row);
        sum.xyz /= 2;
        write_imagef(dst_image, coord, sum);
    }
}
```

//BMP FUNCTIONS

```
unsigned char* read_BMP_RGB_to_RGBA(string filename, int* widthOut, int* heightOut)
{
    char fileHeader[54];    // to store the file header, bmp file format bmpheader (14 bytes) + bmpheaderinfo (40 bytes) =
        54 bytes
    int width, height;      // width and height of image
    int offset;             // offset where image data starts in the file
    int imageSize;          // image size in bytes
    int padSize;            // in the bmp file format, each row must be a multiple of 4
    int rowSize;            // size per row in bytes
    int i, j;
    char colour[3];
    unsigned char* imageData;   // pointer to store image data

    // open file stream
    ifstream textureFileStream(filename, ios::in | ios::binary);

    // check whether file opened successfully
    if (!textureFileStream.is_open())
    {
        cout << "Failed to open texture file - " << filename << endl;
        return NULL;
    }

    // get file header
    textureFileStream.read(fileHeader, 54);

    // get offset, width and height information
    offset = fileHeader[10];
    width = fileHeader[21] << 24;
    width |= fileHeader[20] << 16;
    width |= fileHeader[19] << 8;
    width |= fileHeader[18];
    width = abs(width);
    height = fileHeader[25] << 24;
    height |= fileHeader[24] << 16;
    height |= fileHeader[23] << 8;
    height |= fileHeader[22];
    height = abs(height);

    // allocate RGBA image data
    imageSize = width * height * 4;
    imageData = new unsigned char[imageSize];

    // move read position by offset
    textureFileStream.seekg(offset, ios::beg);
```

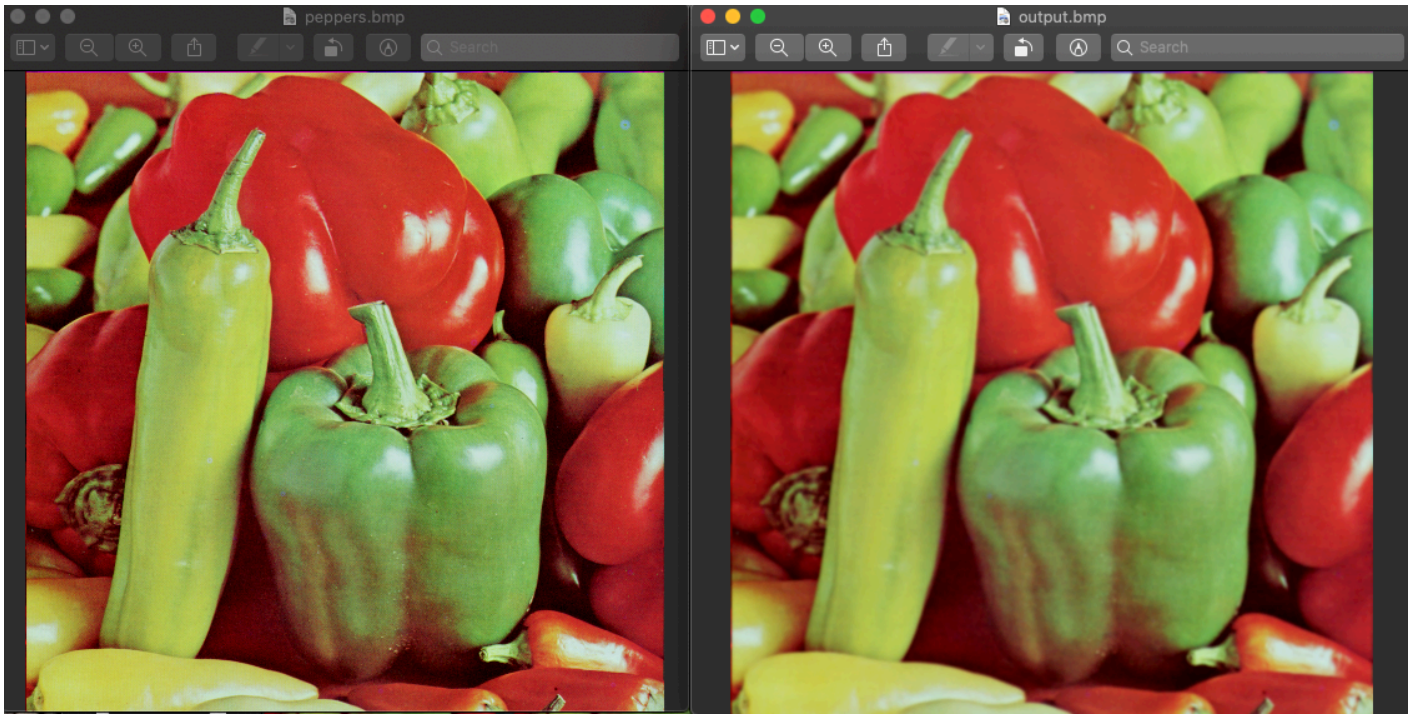
```
    // record width and height, and return pointer to image data
    *widthOut = width;
    *heightOut = height;

    return imageData;
}

// writes imageData (in RGBA format) to a 24-bit RGB bitmap file
void write_BMP_RGBA_to_RGB(string filename, unsigned char* imageData, int width, int height)
{
    char fileHeader[54] = {
        // BITMAPHEADER
        'B','M',        // bmp file
        0, 0, 0, 0,     // file size in bytes
        0, 0,           // reserved
        0, 0,           // reserved
        54, 0, 0, 0,    // offset
        // BITMAPINFOHEADER
        40, 0, 0, 0,    // size of info header
        0, 0, 0, 0,     // width
        0, 0, 0, 0,     // heigth
        1, 0,           // number colour planes
        24, 0,          // number of bits per pixel
        0, 0, 0, 0,     // compression
        0, 0, 0, 0,     // image size
        0, 0, 0, 0,     // horizontal resolution
        0, 0, 0, 0,     // vertical resolution
        0, 0, 0, 0,     // number of colours in palette
        0, 0, 0, 0,     // number of important colours
    };
    int imageSize;      // image size in bytes
    int padSize;        // in the bmp file format, each row must be a multiple of 4
    int rowSize;        // size per row in bytes
    int fileSize;       // file size in bytes (image size + header size)
    int i, j;
    char colour[3];

    // compute amount of padding so that each row is a multiple of 4
    padSize = width % 4;
    if (padSize != 0) {
        padSize = 4 - padSize;
    }

    // compute image size
    imageSize = (width + padSize) * height * 3;

    // open output stream
    ofstream outFileStream(filename, ios::out | ios::binary);
```

```
    // check whether output stream opened successfully
    if (!outFileStream.is_open())
    {
        cout << "Failed to open output file - " << filename << endl;
        return;
    }

    // compute file size (image size + header size)
    fileSize = 54 + imageSize;

    // fill in appropriate bmp header fields in little endian order
    fileHeader[2] = (unsigned char)fileSize;
    fileHeader[3] = (unsigned char)(fileSize >> 8);
    fileHeader[4] = (unsigned char)(fileSize >> 16);
    fileHeader[5] = (unsigned char)(fileSize >> 24);

    fileHeader[18] = (unsigned char)width;
    fileHeader[19] = (unsigned char)(width >> 8);
    fileHeader[20] = (unsigned char)(width >> 16);
    fileHeader[21] = (unsigned char)(width >> 24);

    fileHeader[22] = (unsigned char)height;
    fileHeader[23] = (unsigned char)(height >> 8);
    fileHeader[24] = (unsigned char)(height >> 16);
    fileHeader[25] = (unsigned char)(height >> 24);

    fileHeader[34] = (unsigned char)imageSize;
    fileHeader[35] = (unsigned char)(imageSize >> 8);
    fileHeader[36] = (unsigned char)(imageSize >> 16);
    fileHeader[37] = (unsigned char)(imageSize >> 24);

    // write file header to out stream
    outFileStream.write(fileHeader, 54);

    // bitmaps are stored in upside-down raster order
    rowSize = width * 4;

    // output each RGB pixel colour
    for (i = 0; i < height; i++) {
        for (j = 0; j < rowSize; j += 4) {
            colour[0] = (unsigned char)imageData[i*rowSize + j];
            colour[1] = (unsigned char)imageData[i*rowSize + j + 1];
            colour[2] = (unsigned char)imageData[i*rowSize + j + 2];
            outFileStream.write(colour, 3);
        }

        // in bmp format rows must be a multiple of 4-bytes, add junk padding if required
        for (j = 0; j < padSize; j++) {
            outFileStream.write(colour, 3);
```

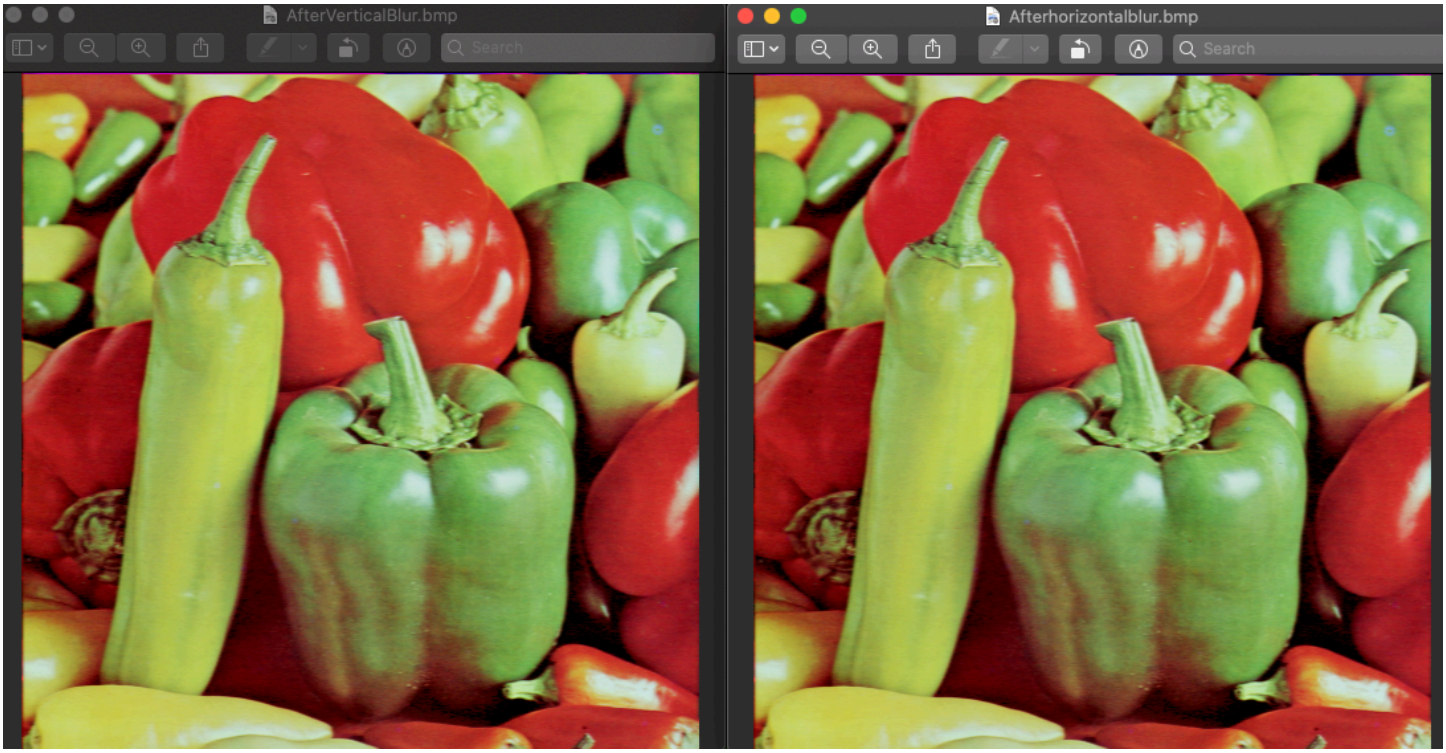
//PROGRAM RUN

```
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: 1
Program build: Successful
--------------------
Select an option:
1. 7x7 window
2. Two-pass
Enter input: 2
Kernel enqueued. (Horizontal)
--------------------
Kernel enqueued. (Vertical)
--------------------
Success.
Program ended with exit code: 0
```

//peppers.bmp (left) //output.bmp (right)



//AfterVerticalBlur.bmp(Left) //AfterHorizontalBlur.bmp(Right)



Task 3c (Bloom effect)

Bloom effects are commonly used in graphics, movies, video games, etc. This part combines the work from Tasks 3a and 3b. The basic steps to create an image with a bloom effect are illustrated as follows:

1. The image in Fig. 1(a) shows the original image

2. The image in Fig. 1(b) shows an image where the glowing pixels are kept, while the rest are 2
   set to black . For this assignment, allow the user to input a valid threshold luminance value. Pixels above the threshold luminance value are kept, while pixels below this luminance value are set to black. This step is related to Task 3a.

3. The image in Fig. 1(b) undergoes a horizontal blur pass, then a vertical blur pass to obtain the image depicted in Fig. 1(c). This step is related to Task 3b.

//Kernel

```
__kernel void blur_image(__global int* option, read_only image2d_t src_image,
    write_only image2d_t dst_image)
{
    //get option chosen by user
    int getOption = option[0];
    int minValue = 0;
    int maxValue = 0;

    // Accumulated pixel value
    float4 sum = (float4)(0.0);                    ⚠ double precision constant requires cl_khr_fp64, casting to single precision

    //Set filter to 0
    int index = 0;

    int2 coord;
    float4 pixel;

        minValue = -3;
        maxValue = 3;

        //Get one direction for pass
        int column = get_global_id(0);
        int row = get_global_id(1);
        coord.y = row;
        index = 0;
        //Iterate over the columns
        for (int j = minValue; j <= maxValue; j++)
        {
            coord.x = column + j;

            //Read pixel from the image
            pixel = read_imagef(src_image, sampler, coord);
            //Acculumate weighted sum

                sum.xyz += pixel.xyz * filterPass7x7[index++];

        }
        //Write pixel to output
        coord = (int2)(column, row);
        write_imagef(dst_image, coord, sum);
    }

__kernel void blur_vertical(__global int* option, read_only image2d_t src_image,
    write_only image2d_t dst_image)
{
    //get option chosen by user
    int getOption = option[0];
    int minValue = 0;
    int maxValue = 0;

    // Accumulated pixel value
    float4 sum = (float4)(0.0);                    ⚠ double precision constant requires cl_khr_fp64, casting to single precision
```

```
    //Set filter to 0
    int index = 0;

    int2 coord;
    float4 pixel;



        minValue = -3;
        maxValue = 3;

    //Get one direction for pass
    int column = get_global_id(0);
    int row = get_global_id(1);

    coord.x = column;
    index = 0;
    //Iterate over the columns
    for (int j = minValue; j <= maxValue; j++)
    {
        coord.y = row + j;

        //Read pixel from the image
        pixel = read_imagef(src_image, sampler, coord);
        //Acculumate weighted sum

        //Write pixel to output
        coord = (int2)(column, row);
        write_imagef(dst_image, coord, sum);
    }
}
```

//BMP FUNCTIONS

```cpp
unsigned char* read_BMP_RGB_to_RGBA(string filename, int* widthOut, int* heightOut)
{
    char fileHeader[54];    // to store the file header, bmp file format bmpheader (14 bytes) + bmpheaderinfo (40 bytes) = 54 bytes
    int width, height;      // width and height of image
    int offset;             // offset where image data starts in the file
    int imageSize;          // image size in bytes
    int padSize;            // in the bmp file format, each row must be a multiple of 4
    int rowSize;            // size per row in bytes
    int i, j;
    char colour[3];
    unsigned char* imageData;   // pointer to store image data

    // open file stream
    ifstream textureFileStream(filename, ios::in | ios::binary);

    // check whether file opened successfully
    if (!textureFileStream.is_open())
    {
        cout << "Failed to open texture file - " << filename << endl;
        return NULL;
    }

    // get file header
    textureFileStream.read(fileHeader, 54);

    // get offset, width and height information
    offset = fileHeader[10];
    width = fileHeader[21] << 24;
    width |= fileHeader[20] << 16;
    width |= fileHeader[19] << 8;
    width |= fileHeader[18];
    width = abs(width);
    height = fileHeader[25] << 24;
    height |= fileHeader[24] << 16;
    height |= fileHeader[23] << 8;
    height |= fileHeader[22];
    height = abs(height);

    // allocate RGBA image data
    imageSize = width * height * 4;
    imageData = new unsigned char[imageSize];

    // move read position by offset
    textureFileStream.seekg(offset, ios::beg);

    // compute amount of row padding
    padSize = width % 4;
    if (padSize != 0) {
        padSize = 4 - padSize;
    }
```

```cpp
    // bitmaps are stored in upside-down raster order
    rowSize = width * 4;

    // read each RGB pixel colour
    for (i = 0; i < height; i++) {
        for (j = 0; j < rowSize; j += 4) {
            textureFileStream.read(colour, 3);
            imageData[i*rowSize + j] = colour[0];
            imageData[i*rowSize + j + 1] = colour[1];
            imageData[i*rowSize + j + 2] = colour[2];
            imageData[i*rowSize + j + 3] = 255;
        }
        // in the bmp format, each row has to be a multiple of 4
        // read in the junk data and throw it away
        for (j = 0; j < padSize; j++) {
            textureFileStream.read(colour, 3);
        }
    }

    // close file stream
    textureFileStream.close();

    // record width and height, and return pointer to image data
    *widthOut = width;
    *heightOut = height;

    return imageData;
}
```

```cpp
void GetGlowingPixels(string filename, float average)
{
    char fileHeader[54];
    int width, height;
    int offset, imageSize, fileSize;
    char colour[3];
    unsigned char* imageData;

    // open image file
    ifstream getImageFile(filename, ios::in | ios::binary);
    // check whether file opened successfully
    if (getImageFile.is_open() == false)
    {
        cout << "Failed to open texture file - " << filename << endl;
    }
    //process file header
    getImageFile.read(fileHeader, 54);

    //get information for offset, width and height
    offset = fileHeader[10];
    width = fileHeader[21] << 24;
    width |= fileHeader[20] << 16;
    width |= fileHeader[19] << 8;
    width |= fileHeader[18];
    width = abs(width);
    height = fileHeader[25] << 24;
    height |= fileHeader[24] << 16;
    height |= fileHeader[23] << 8;
    height |= fileHeader[22];
    height = abs(height);

    // compute amount of row padding

    //create imageData to store RGB values of each pixels
    imageData = new unsigned char[width*height * 3];
    //get each rgb value
    int rgbRow = width * 3;
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < rgbRow; j += 3) {
            getImageFile.read(colour, 3);
            imageData[i*rgbRow + j] = colour[0];
            imageData[i*rgbRow + j + 1] = colour[1];
            imageData[i*rgbRow + j + 2] = colour[2];
        }
    }
    //close file
    getImageFile.close();

    //create file to be exported
    ofstream outFileStream("/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/glow.bmp", ios::out | ios::binary);
    if (outFileStream.is_open() == false)
    {
        cout << "Failed to open output file - glowImage.bmp" << endl;
    }
    //get size of RGB information
    imageSize = width * height * 3;
    //compute file size
    fileSize = 54 + imageSize;

    //fill in appropriate bmp header fields
    fileHeader[2] = (unsigned char)fileSize;
    fileHeader[3] = (unsigned char)(fileSize >> 8);
    fileHeader[4] = (unsigned char)(fileSize >> 16);
    fileHeader[5] = (unsigned char)(fileSize >> 24);

    fileHeader[18] = (unsigned char)width;
    fileHeader[19] = (unsigned char)(width >> 8);
    fileHeader[20] = (unsigned char)(width >> 16);
    fileHeader[21] = (unsigned char)(width >> 24);

    fileHeader[22] = (unsigned char)height;
    fileHeader[23] = (unsigned char)(height >> 8);
    fileHeader[24] = (unsigned char)(height >> 16);
    fileHeader[25] = (unsigned char)(height >> 24);

    fileHeader[34] = (unsigned char)imageSize;
    fileHeader[35] = (unsigned char)(imageSize >> 8);
    fileHeader[36] = (unsigned char)(imageSize >> 16);
    fileHeader[37] = (unsigned char)(imageSize >> 24);

    //write file
    outFileStream.write(fileHeader, 54);
    int r = 0, g = 0, b = 0;
    float tempLuminance = 0.0;
    //set each RGB pixel colour to image
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < rgbRow; j += 3) {
            //get rgb information
            r = imageData[i*rgbRow + j];
            g = imageData[i*rgbRow + j + 1];
            b = imageData[i*rgbRow + j + 2];
            //using luminance formula '0.299*R + 0.587*G + 0.114*B' then set each rgb value to the luminance value
            tempLuminance = (float)(0.299*r + 0.587*g + 0.114*b);
            //change pixel to black if luminance value less than average value
            if (tempLuminance < (int)average)
            {
                colour[0] = 0;
                colour[1] = 0;
                colour[2] = 0;
                outFileStream.write(colour, 3);
            }
```

```cpp
void write_BMP_RGBA_to_RGB(string filename, unsigned char* imageData, int width, int height)
{
    char fileHeader[54] = {
        // BITMAPHEADER
        'B','M',        // bmp file
        0, 0, 0, 0,     // file size in bytes
        0, 0,           // reserved
        0, 0,           // reserved
        54, 0, 0, 0,    // offset
        // BITMAPINFOHEADER
        40, 0, 0, 0,    // size of info header
        0, 0, 0, 0,     // width
        0, 0, 0, 0,     // heigth
        1, 0,           // number colour planes
        24, 0,          // number of bits per pixel
        0, 0, 0, 0,     // compression
        0, 0, 0, 0,     // image size
        0, 0, 0, 0,     // horizontal resolution
        0, 0, 0, 0,     // vertical resolution
        0, 0, 0, 0,     // number of colours in palette
        0, 0, 0, 0,     // number of important colours
    };
    int imageSize;      // image size in bytes
    int padSize;        // in the bmp file format, each row must be a multiple of 4
    int rowSize;        // size per row in bytes
    int fileSize;       // file size in bytes (image size + header size)
    int i, j;
    char colour[3];

    // compute amount of padding so that each row is a multiple of 4
    padSize = width % 4;
    if (padSize != 0) {
        padSize = 4 - padSize;
    }

    // compute image size
    imageSize = (width + padSize) * height * 3;

    // open output stream
    ofstream outFileStream(filename, ios::out | ios::binary);

    // check whether output stream opened successfully
    if (!outFileStream.is_open())
    {
        cout << "Failed to open output file - " << filename << endl;
        return;
    }

    // compute file size (image size + header size)
    fileSize = 54 + imageSize;

    // fill in appropriate bmp header fields in little endian order
    fileHeader[2] = (unsigned char)fileSize;
    fileHeader[3] = (unsigned char)(fileSize >> 8);
    fileHeader[4] = (unsigned char)(fileSize >> 16);
    fileHeader[5] = (unsigned char)(fileSize >> 24);

    fileHeader[18] = (unsigned char)width;
    fileHeader[19] = (unsigned char)(width >> 8);
    fileHeader[20] = (unsigned char)(width >> 16);
    fileHeader[21] = (unsigned char)(width >> 24);

    fileHeader[22] = (unsigned char)height;
    fileHeader[23] = (unsigned char)(height >> 8);
    fileHeader[24] = (unsigned char)(height >> 16);
    fileHeader[25] = (unsigned char)(height >> 24);

    fileHeader[34] = (unsigned char)imageSize;
    fileHeader[35] = (unsigned char)(imageSize >> 8);
    fileHeader[36] = (unsigned char)(imageSize >> 16);
    fileHeader[37] = (unsigned char)(imageSize >> 24);

    // write file header to out stream
    outFileStream.write(fileHeader, 54);

    // bitmaps are stored in upside-down raster order
    rowSize = width * 4;

    // output each RGB pixel colour
    for (i = 0; i < height; i++) {
        for (j = 0; j < rowSize; j += 4) {
            colour[0] = (unsigned char)imageData[i*rowSize + j];
            colour[1] = (unsigned char)imageData[i*rowSize + j + 1];
            colour[2] = (unsigned char)imageData[i*rowSize + j + 2];
            outFileStream.write(colour, 3);
        }

        // in bmp format rows must be a multiple of 4-bytes, add junk padding if required
        for (j = 0; j < padSize; j++) {
            outFileStream.write(colour, 3);
        }
    }

    // close output file stream
    outFileStream.close();
}
```

```cpp
//combine pixels of both image for glow effect
void AddPixelsTogether(string originalFile, string toAdd)
{
    char fileHeader[54];
    int width, height;
    int offset, imageSize, fileSize;
    char colour[3];
    unsigned char *imageData, *addData;

    // open image file
    ifstream getImageFile(originalFile, ios::in | ios::binary);
    // check whether file opened successfully
    if (getImageFile.is_open() == false)
    {
        cout << "Failed to open texture file - " << originalFile << endl;
    }
    //process file header
    getImageFile.read(fileHeader, 54);

    //get information for offset, width and height
    offset = fileHeader[10];
    width = fileHeader[21] << 24;
    width |= fileHeader[20] << 16;
    width |= fileHeader[19] << 8;
    width |= fileHeader[18];
    width = abs(width);
    height = fileHeader[25] << 24;
    height |= fileHeader[24] << 16;
    height |= fileHeader[23] << 8;
    height |= fileHeader[22];
    height = abs(height);

    //create imageData to store RGB values of each pixels
    imageData = new unsigned char[width*height * 3];
    //get each rgb value
    int rgbRow = width * 3;
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < rgbRow; j += 3) {
            getImageFile.read(colour, 3);
            imageData[i*rgbRow + j] = colour[0];
            imageData[i*rgbRow + j + 1] = colour[1];
            imageData[i*rgbRow + j + 2] = colour[2];
        }
    }
    //close file
    getImageFile.close();

    // open second image file
    ifstream getAddFile(toAdd, ios::in | ios::binary);
    // check whether file opened successfully
    if (getAddFile.is_open() == false)
    {
        cout << "Failed to open texture file - " << toAdd << endl;
    }
    //process file header
    getAddFile.read(fileHeader, 54);
    //create imageData to store RGB values of each pixels
    addData = new unsigned char[width*height * 3];
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < rgbRow; j += 3) {
            getAddFile.read(colour, 3);
            addData[i*rgbRow + j] = colour[0];
            addData[i*rgbRow + j + 1] = colour[1];
            addData[i*rgbRow + j + 2] = colour[2];
        }
    }
    //close file
    getAddFile.close();


    //create file to be exported
    ofstream outFileStream("/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/final.bmp", ios::out | ios::binary);
    if (outFileStream.is_open() == false)
    {
        cout << "Failed to open output file - final.bmp" << endl;
    }
    //get size of RGB information
    imageSize = width * height * 3;
    //compute file size
    fileSize = 54 + imageSize;

    //fill in appropriate bmp header fields
    fileHeader[2] = (unsigned char)fileSize;
    fileHeader[3] = (unsigned char)(fileSize >> 8);
    fileHeader[4] = (unsigned char)(fileSize >> 16);
    fileHeader[5] = (unsigned char)(fileSize >> 24);

    fileHeader[18] = (unsigned char)width;
    fileHeader[19] = (unsigned char)(width >> 8);
    fileHeader[20] = (unsigned char)(width >> 16);
    fileHeader[21] = (unsigned char)(width >> 24);

    fileHeader[22] = (unsigned char)height;
    fileHeader[23] = (unsigned char)(height >> 8);
    fileHeader[24] = (unsigned char)(height >> 16);
    fileHeader[25] = (unsigned char)(height >> 24);

    fileHeader[34] = (unsigned char)imageSize;
    fileHeader[35] = (unsigned char)(imageSize >> 8);
    fileHeader[36] = (unsigned char)(imageSize >> 16);
    fileHeader[37] = (unsigned char)(imageSize >> 24);
```

```
399     //write file
400     outFileStream.write(fileHeader, 54);
401     int r = 0, g = 0, b = 0;
402     //set each RGB pixel colour to image
403     for (int i = 0; i < height; i++) {
404         for (int j = 0; j < rgbRow; j += 3) {
405             //get rgb information and add the pixels together
406             r = imageData[i*rgbRow + j] + addData[i*rgbRow + j];
407             g = imageData[i*rgbRow + j + 1] + addData[i*rgbRow + j + 1];
408             b = imageData[i*rgbRow + j + 2] + addData[i*rgbRow + j + 2];
409             //keep max value to 255 if the sum exceeds
410             if (r > 255)
411             {
412                 colour[0] = 255;                    ⚠ Implicit conversion from 'int' to 'char' changes value from 255 to -1
413             }
414             else
415             {
416                 colour[0] = r;
417             }
418
419             if (g > 255)
420             {
421                 colour[1] = 255;                    ⚠ Implicit conversion from 'int' to 'char' changes value from 255 to -1
422             }
423             else
424             {
425                 colour[1] = g;
426             }
427             if (b > 255)
428             {
429                 colour[2] = 255;                    ⚠ Implicit conversion from 'int' to 'char' changes value from 255 to -1
430             }
431             else
432             {
433                 colour[2] = b;
434             }
435             outFileStream.write(colour, 3);
436         }
437     }
438
439     outFileStream.close();
440     delete[] imageData;
441     delete[] addData;
442 }
443
```

//.cpp file (Buffer, Enqueue Kernel)

```
52      try
53      {
54          if (!select_one_device(&platform, &devices))
55          {
56              // if no device selected
57              quit_program("Device not selected.");
58          }
59          else
60          {
61              //create details
62              details = Context(devices);
63              cout << endl;
64              cout << "Checking if programming is built successfully..." << endl;
65              bool programBuilt = build_program(&programming, &details, "/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/Task3c.cl");
66              if (programBuilt == false)
67              {
68                  // if OpenCL program build error
69                  quit_program("OpenCL program build error.");
70              }
71              else
72              {
73                  //-----user to choose which filter option to use -----//
74                  //for final choice
75                  int option = 0;
76                  //variable to contain temporary choice from user
77                  string choice = "0";
78
79                  string fileName = "/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/peppers.bmp";
80
81                  //average luminance value from part 2
82                  float luminanceThreshold = 0;
83                  cout<< "Enter luminance Treshold: ";
84                  cin>>luminanceThreshold;
85
86
87                  //keep pixel colour if above average luminance value, else change to black
88                  GetGlowingPixels(fileName, luminanceThreshold);
89                  cout << " image showing the glowing pixels exported" << endl;
90
91                  //perform horizontal then vertical blur pass and export image-----
92                  //create kernel
93                  kernel[BLUR_IMAGE] = Kernel(programming, "blur_image");
94                  kernel[BLUR_VERTICAL] = Kernel(programming, "blur_vertical");
95
96                  //create command queue and enable profiling
97                  queue = CommandQueue(details, devices);
98                  inImage = read_BMP_RGB_to_RGBA("/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/glow.bmp", &width, &height);
99
100                 //allocate memory
101                 int size = width * height * 4;
102                 outImage = new unsigned char[size];
```

```
                //allocate memory
                int size = width * height * 4;
                outImage = new unsigned char[size];

                imageFormat = ImageFormat(CL_RGBA, CL_UNORM_INT8);
                //create image object
                optionBuffer = Buffer(details, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(cl_int) * 1, &option);
                inImageBuffer = Image2D(details, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, imageFormat, width, height, 0, (void*)inImage);
                outImageBuffer = Image2D(details, CL_MEM_WRITE_ONLY | CL_MEM_COPY_HOST_PTR, imageFormat, width, height, 0, (void*)outImage);

                // set kernel arguments
                kernel[BLUR_IMAGE].setArg(0, optionBuffer);
                kernel[BLUR_IMAGE].setArg(1, inImageBuffer);
                kernel[BLUR_IMAGE].setArg(2, outImageBuffer);

                //read image from device to host memory
                cl::size_t<3> origin, region;
                origin[0] = origin[1] = origin[2] = 0;
                region[0] = width;
                region[1] = height;
                region[2] = 1;

                NDRange offset(0, 0);
                NDRange globalSize(width, height);
                //initialize timeTotal to 0

                //horizontal pass
                queue.enqueueNDRangeKernel(kernel[BLUR_IMAGE], offset, globalSize, NullRange);
                queue.enqueueReadImage(outImageBuffer, CL_TRUE, origin, region, 0, 0, outImage);

                    //get all after horizontal pass image
                    write_BMP_RGBA_to_RGB("/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/Afterhorizontalblur.bmp", outImage, width, height);
                    cout << "After horizontal blur image exported" << endl;
                    free(inImage);
                    free(outImage);

                    //get after vertical pass image
                    inImage = read_BMP_RGB_to_RGBA("/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/Afterhorizontalblur.bmp", &width, &height);
                    outImage = new unsigned char[size];
                    inImageBuffer = Image2D(details, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, imageFormat, width, height, 0, (void*)inImage);
                    kernel[BLUR_VERTICAL].setArg(0, optionBuffer);
                    kernel[BLUR_VERTICAL].setArg(1, inImageBuffer);
                    kernel[BLUR_VERTICAL].setArg(2, outImageBuffer);

                    queue.enqueueNDRangeKernel(kernel[BLUR_VERTICAL], offset, globalSize, NullRange);
                    queue.enqueueReadImage(outImageBuffer, CL_TRUE, origin, region, 0, 0, outImage);
                    write_BMP_RGBA_to_RGB("/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/Afterverticalblur.bmp", outImage, width, height);
                    cout << "After vertical blur image exported." << endl;

                     //get bloom effect
                     AddPixelsTogether(fileName, "/Users/isaacyeo/Desktop/A2Task3C/A2Task3C/Afterverticalblur.bmp");
                     cout << "final.bmp exported" << endl;
                     free(inImage);
                     free(outImage);









            }
        }
        cout << endl;

    }
    catch (Error e)
    {
        handle_error(e);
        cout << e.err() << " (" << lookup_error_code(e.err()) << ")" << endl;
    }
    system("Pause");
    return 0;
}

```

```
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: 1

Checking if programming is built successfully...
Program build: Successful
--------------------
Enter luminance Treshold: 150
 image showing the glowing pixels exported
After horizontal blur image exported
After vertical blur image exported.
final.bmp exported
```