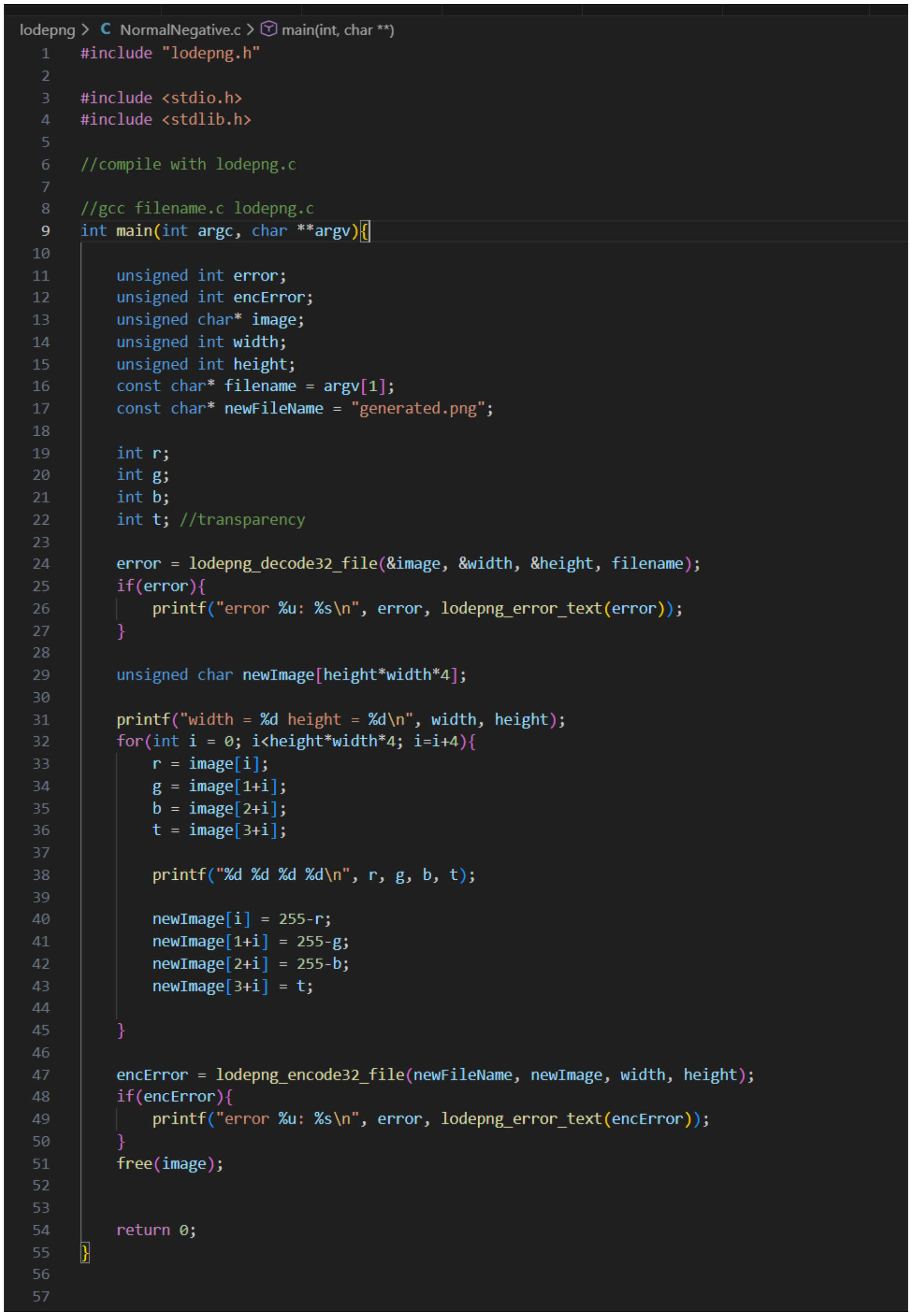
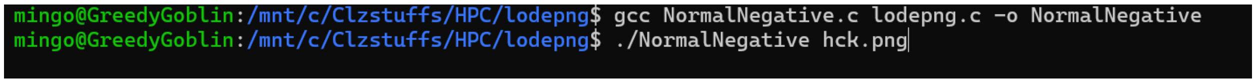
NormalNegative



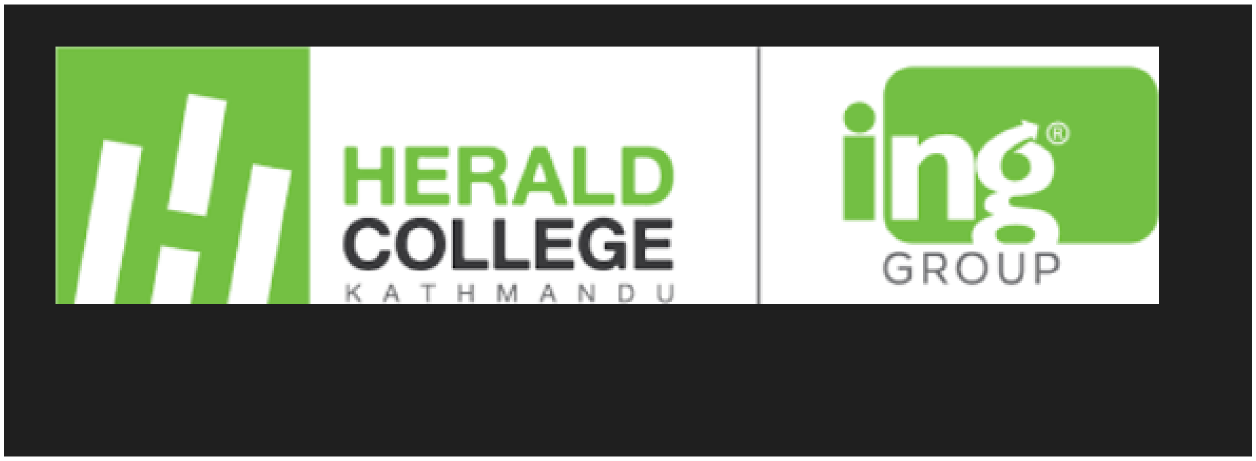
lodepng > C NormalNegative.c > main(int, char **)

```c
#include "lodepng.h"

#include <stdio.h>
#include <stdlib.h>

//compile with lodepng.c

//gcc filename.c lodepng.c
int main(int argc, char **argv){

    unsigned int error;
    unsigned int encError;
    unsigned char* image;
    unsigned int width;
    unsigned int height;
    const char* filename = argv[1];
    const char* newFileName = "generated.png";

    int r;
    int g;
    int b;
    int t; //transparency

    error = lodepng_decode32_file(&image, &width, &height, filename);
    if(error){
        printf("error %u: %s\n", error, lodepng_error_text(error));
    }

    unsigned char newImage[height*width*4];

    printf("width = %d height = %d\n", width, height);
    for(int i = 0; i<height*width*4; i=i+4){
        r = image[i];
        g = image[1+i];
        b = image[2+i];
        t = image[3+i];

        printf("%d %d %d %d\n", r, g, b, t);

        newImage[i] = 255-r;
        newImage[1+i] = 255-g;
        newImage[2+i] = 255-b;
        newImage[3+i] = t;

    }

    encError = lodepng_encode32_file(newFileName, newImage, width, height);
    if(encError){
        printf("error %u: %s\n", error, lodepng_error_text(encError));
    }
    free(image);


    return 0;
}
```

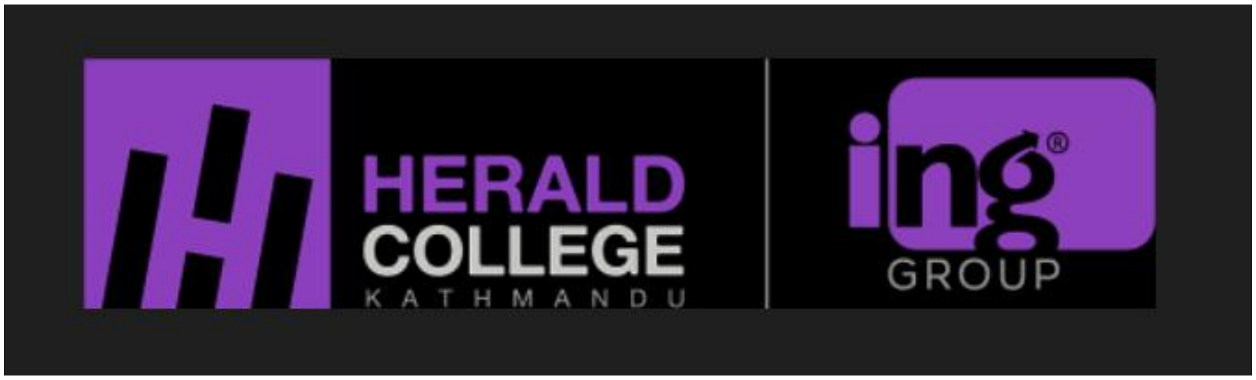


```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/lodepng$ gcc NormalNegative.c lodepng.c -o NormalNegative
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/lodepng$ ./NormalNegative hck.png
```

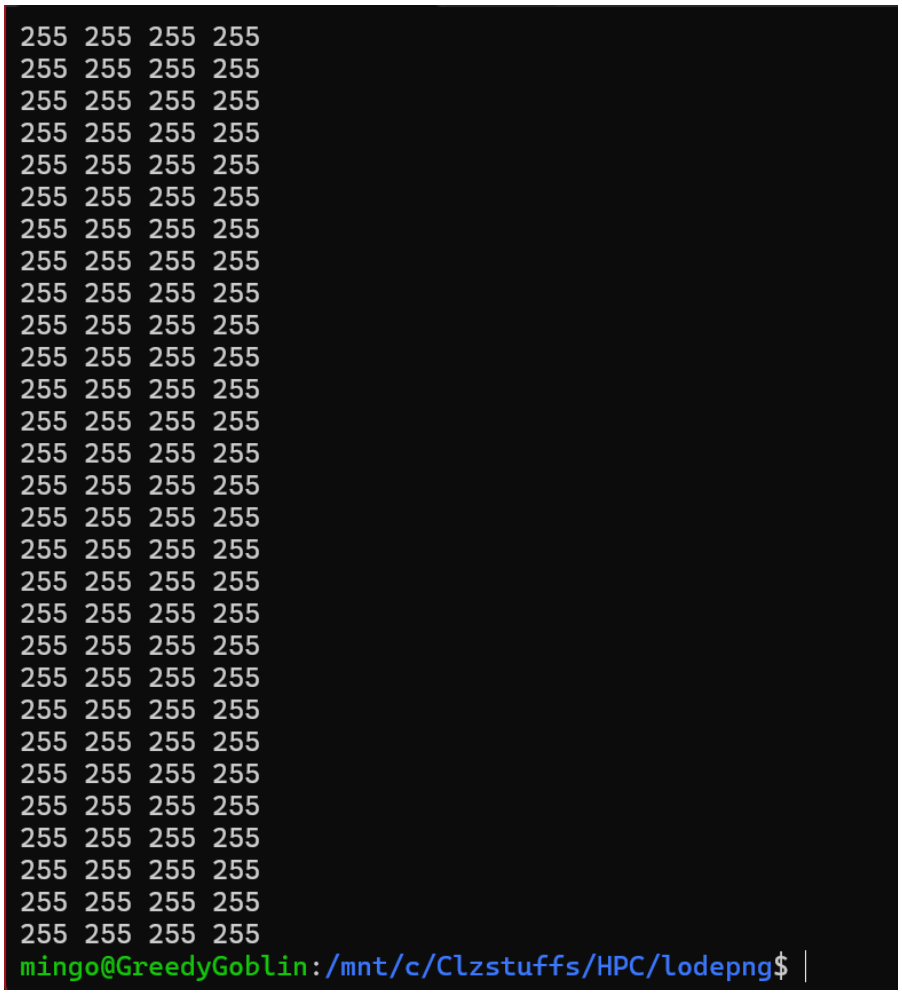
Input



Output

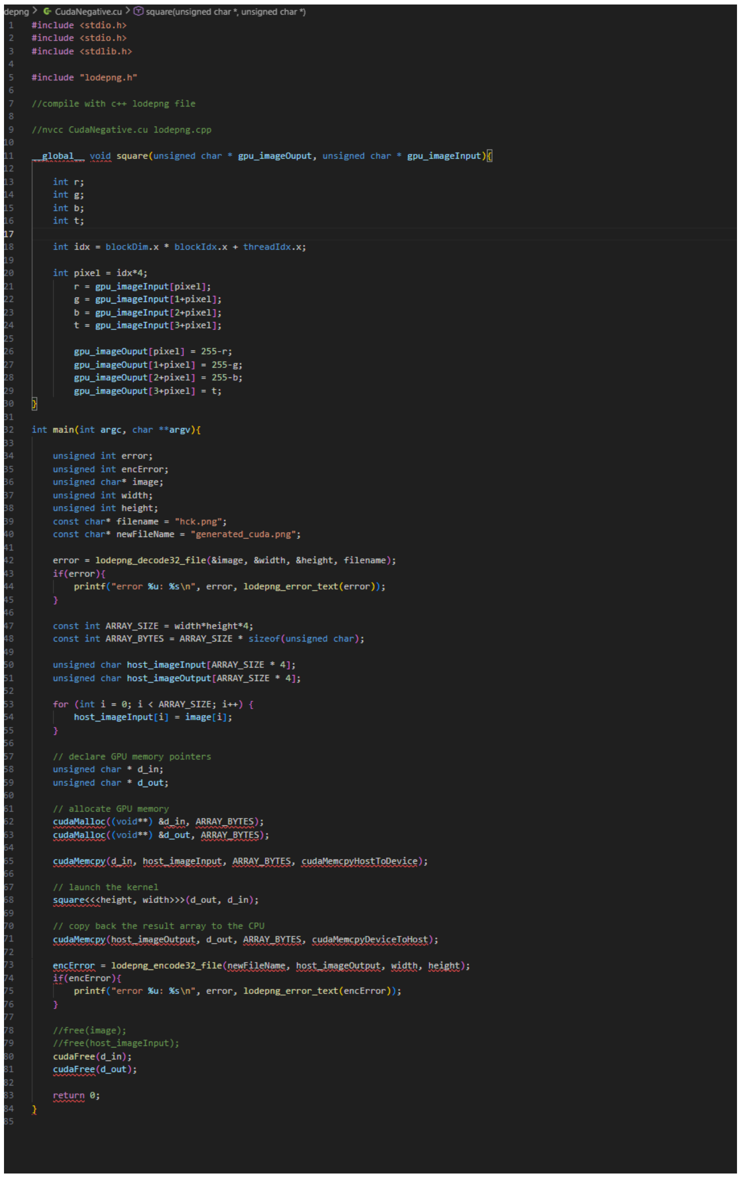


This program loads a PNG image with the Lodepng library and inverts the color values of the image giving a negative effect, and saves the result as a different PNG file named generated.png. It keeps the alpha channel intact but each pixel in RGBA format is processed. The input PNG file is accepted as a command-line argument by the program followed by the process of decoding the image to the memory and later inverting the colors and saving the inverted image back.



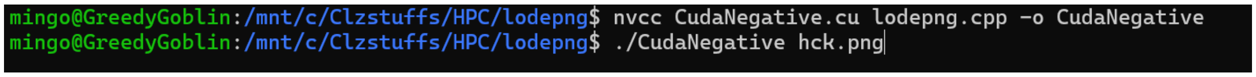
```
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
255 255 255 255
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/lodepng$
```

CudaNegative



depng > CudaNegative.cu > square(unsigned char *, unsigned char *)

```c
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>

#include "lodepng.h"

//compile with c++ lodepng file

//nvcc CudaNegative.cu lodepng.cpp

__global__ void square(unsigned char * gpu_imageOuput, unsigned char * gpu_imageInput){

    int r;
    int g;
    int b;
    int t;

    int idx = blockDim.x * blockIdx.x + threadIdx.x;

    int pixel = idx*4;
        r = gpu_imageInput[pixel];
        g = gpu_imageInput[1+pixel];
        b = gpu_imageInput[2+pixel];
        t = gpu_imageInput[3+pixel];

        gpu_imageOuput[pixel] = 255-r;
        gpu_imageOuput[1+pixel] = 255-g;
        gpu_imageOuput[2+pixel] = 255-b;
        gpu_imageOuput[3+pixel] = t;
}

int main(int argc, char **argv){

    unsigned int error;
    unsigned int encError;
    unsigned char* image;
    unsigned int width;
    unsigned int height;
    const char* filename = "hck.png";
    const char* newFileName = "generated_cuda.png";

    error = lodepng_decode32_file(&image, &width, &height, filename);
    if(error){
        printf("error %u: %s\n", error, lodepng_error_text(error));
    }

    const int ARRAY_SIZE = width*height*4;
    const int ARRAY_BYTES = ARRAY_SIZE * sizeof(unsigned char);

    unsigned char host_imageInput[ARRAY_SIZE * 4];
    unsigned char host_imageOutput[ARRAY_SIZE * 4];

    for (int i = 0; i < ARRAY_SIZE; i++) {
        host_imageInput[i] = image[i];
    }

    // declare GPU memory pointers
    unsigned char * d_in;
    unsigned char * d_out;

    // allocate GPU memory
    cudaMalloc((void**) &d_in, ARRAY_BYTES);
    cudaMalloc((void**) &d_out, ARRAY_BYTES);

    cudaMemcpy(d_in, host_imageInput, ARRAY_BYTES, cudaMemcpyHostToDevice);

    // launch the kernel
    square<<<height, width>>>(d_out, d_in);

    // copy back the result array to the CPU
    cudaMemcpy(host_imageOutput, d_out, ARRAY_BYTES, cudaMemcpyDeviceToHost);

    encError = lodepng_encode32_file(newFileName, host_imageOutput, width, height);
    if(encError){
        printf("error %u: %s\n", error, lodepng_error_text(encError));
    }

    //free(image);
    //free(host_imageInput);
    cudaFree(d_in);
    cudaFree(d_out);

    return 0;
}
```

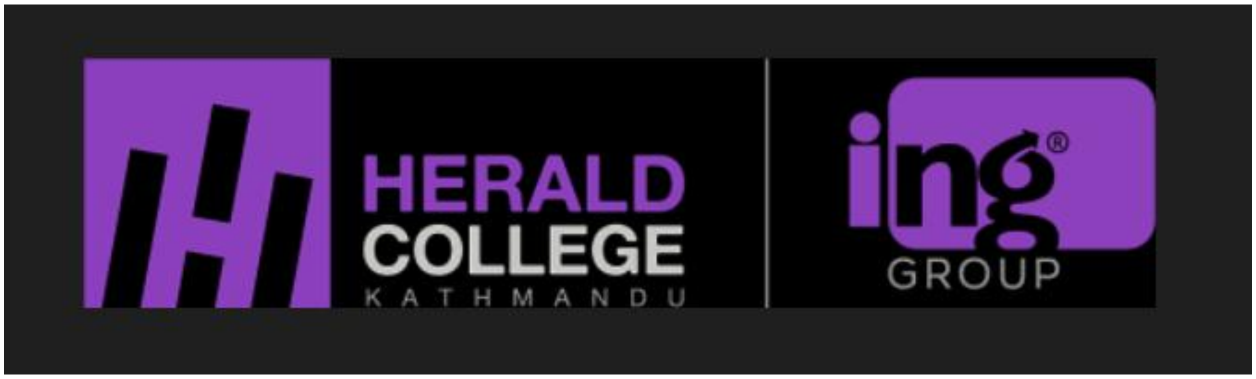


```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/lodepng$ nvcc CudaNegative.cu lodepng.cpp -o CudaNegative
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/lodepng$ ./CudaNegative hck.png
```

Input

HERALD
COLLEGE
KATHMANDU
ing
GROUP

Output



It is a program that takes a PNG image and loads it with the help of Lodepng and then flips the color RGB values to the inverted form, maintaining the alpha channel of the image. The CPU is tasked with reading image off disk, allocating host memory and writing final PNG file. The square kernel is used to invert the color per-pixel of pixels and the colors are inverted parallel using the square kernel with one thread per pixel. The copying of memory to the CPU and its copying to the GPU are performed before the kernel executes and after that. This application of CUDA enables the color inversion process to take a lot less time than would otherwise take place on the CPU alone, particularly to large images.