ASSIGNMENT-1

2.1

For the basic boot loader, firstly I have implemented functions to retrieve the file handle and volume handle to the kernel file.

I'm passing the file handle to my LoadKernel() function which is allocating a buffer using AllocatePool() and reading the kernel file into the buffer.
Now CloseKernel() is called to close the file handle and volume handle.

Next, moving on to the graphics setup – I'm calling my function SetGraphicsMode() and passing the required width and height of 800 and 600 respectively.

Inside SetGraphicsMode(), I'm querying each mode and checking whether the pixel format is BlueGreenRedReserved and the horizontal and vertical resolution as well. If the condition matches, I'm using SetMode() to set the mode otherwise, continue to next iteration.

I'm returning a pointer to the FrameBufferBase back to the main function.

Final step in the boot.c file is to call the kernel function and passing the FrameBufferBase pointer and the dimensions of the framebuffer.

Next, in the kernel.c file I'm running a loop with the formula: fb[i + j*width] = color, where i=width and j=height.

2.2

For this question, we need to implement memory map and ExitBootServices functionality.

There won't be any change in code for kernel.c
In boot.c – in the main function I'm calling setExitBootServices() function right before jumping to the kernel.
Inside this function, I'm getting the memory map, allocating a buffer to store it using the EfiLoaderData.

Using the mapKey received from the previous step and the image handle, I'm calling the ExitBootServices.

2.3

For the last part, we have to implement paging.
Called AllocatePages() and passing the buffer to the kernel file.

In kernel.c file, we set up the page table, page directory, page directory pointer table, and pml4 table.
Then we pass the base address of the pml4 table to the cr3 register.
Moved the frame buffer function to the end of the code after setting up the page tables.

Output Screenshot:

I’ve run this on virtual machine – enabling uefi and allocating 1GB memory.