**BLOCKAPPS ASSIGNMENT**

**PROBLEM STATEMENT:**

Write a haskell program that can create the photo negative from an image stored in the BMP File

Format. The output should also be in BMP format.

**ASSUMPTIONS:**

* No compression was used on the image
* Bits per pixel = 24

**TOOL USAGE:**

This tool provides a command-line interface for running the program with input and output file paths.

1. Compile the Haskell code using GHC.

A black background with white text

Description automatically generated with low confidence

A black background with white text

Description automatically generated with low confidence

**PhotoNegative.hs –** Source file to convert input bmp image to photo negative image

**PhotoNegative –** Module exposing main function and functions required for unit testing

2. Run the compiled executable with command-line arguments

Help option shows the command line arguments required

A picture containing text, font, screenshot, line

Description automatically generated

**input\_file.bmp** – Path of BMP image

**output\_file.bmp –** Path where photo negative BMP image should be created

Below is the sample execution result

A black screen with white text

Description automatically generated with low confidence

3. The program will perform the following steps:

- Read the input BMP file as a byte string.

- Parse the BMP header to check for file format, bits per pixel, and compression.

- If the file format is BMP and the bits per pixel is 24, continue; otherwise, display an error message and exit.

- If the image is compressed, display an error message and exit.

- Create the photo negative from the image data.

- Write the photo negative to the output BMP file.

- Display a success message if the process is completed successfully.

**TYPES:**

- **BMPHeader**: Represents the BMP header structure containing bitsPerPixel(Word16) and compression(Word32) fields.

**CONSTANTS:**

- **headerSize :: Int64**: Represents the size of the BMP header in bytes.

**BMP HEADER PARSING:**

Header size is set to 54 bytes as **BITMAPFILEHEADER** contains 14 bytes followed by **BITMAPINFOHEADER** which contains 40 bytes. First 2 bytes from **BITMAPFILEHEADER** is used to check file format. **biBitCount** field from **BITMAPINFOHEADER** is used to fetch bits per pixel value. **biCompression** field from **BITMAPINFOHEADER**  is used to check for image compression.

Reference links - <https://learn.microsoft.com/en-us/windows/win32/api/wingdi/ns-wingdi-bitmapfileheader>

<https://learn.microsoft.com/en-us/windows/win32/api/wingdi/ns-wingdi-bitmapinfoheader>

**UTILITY FUNCTIONS:**

- **createPhotoNegative :: B.ByteString -> B.ByteString**: Takes a byte string representing the image data and returns a byte string representing the photo negative of the image.

- **isCompressed :: BMPHeader -> Bool**: Takes a BMPHeader and returns True if the image is compressed, False otherwise.

- **isBMPFormat :: B.ByteString** -> Bool: Takes a byte string representing the image content and returns True if the file format is BMP, False otherwise.

- **isValidBitsPerPixel :: BMPHeader -> Bool**: Takes a BMPHeader and returns True if the bits per pixel is 24, False otherwise.

- **invertPixel :: Word8 -> Word8**: Takes a single pixel value (Word8) and returns its complement.

**FUNCTIONS FOR PARSING AND PROCESSING BMP FILES:**

- **parseBMPHeaderFromBytes :: B.ByteString -> Either String BMPHeader:** Takes a byte string representing the image content and parses the BMP header, returning either an error message (Left String) or the parsed BMPHeader(Right BMPHeader).

- **processBMPFile :: B.ByteString -> FilePath -> IO ():** Takes the image content as a byte string and the output file path, extracts the image data, creates the photo negative, and writes it to the output file.

**MAIN FUNCTION:**

- `main :: IO ()`: The main entry point of the program. It reads the command line arguments, handles the `--help` option, reads the input BMP file as a byte string, parses the BMP header, checks the file format, bits per pixel, and compression, and processes the BMP file to create the photo negative.

**SAMPLE INPUT/OUTPUT:**

1. **INPUT : RGB BMP IMAGE**

**A picture containing colorfulness, graphics, red, pattern

Description automatically generated**

**OUTPUT : NEGATIVE RGB BMP IMAGE**

**A picture containing colorfulness, graphics, lilac

Description automatically generated**

1. **INPUT : BLACKBUCK BMP IMAGE**

****

**OUTPUT : NEGATIVE BLACKBUCK BMP IMAGE**

**A picture containing mammal, antelope, antler, wildlife

Description automatically generated**

1. **INPUT : ALL COLOR BMP IMAGE**

**A picture containing sky, tree, outdoor, landscape

Description automatically generated**

**OUTPUT : NEGATIVE ALL COLOR BMP IMAGE**

**A picture containing snow, cloud, painting, landscape

Description automatically generated**

1. **MONOCHROMATIC IMAGE:** Monochromatic image will not be processed as bits per pixel is 8 since only images with 24 bits per pixel will be processed.

**A black screen with white text

Description automatically generated with low confidence**

**BMP PHOTO NEGATIVE VERIFICATION:**

Validated the output by comparing it with online BMP to photo negative conversion tool.

<https://www.imgonline.com.ua/eng/make-negative.php>

**UNIT TESTING:**

Used Cabal command line tool to install the Hunit library

**cabal install HUnit --lib**

**PhotoNegativetests.hs –** Written unit tests for utility functions that are exposed from source file(PhotoNegative.hs) using Hunit library.

- Compile and run the unit tests using following commands