ESE5320 Homework 2

Rico Zhuang

September 12, 2025

Answers

1. **Identify**

   1) **Scale**: this function downscales an image(stored in a 1d array "input") by only outputing every two pixel.

      Filter: applies a convolution filter to the image; first horizontally, then vertically, using hard-coded coefficients. It takes the input image to produce a smoothed output image.

      Differentiate: for each pixel, use the value of its left pixel and its up pixel to calculate an average, and output will be the difference between that pixel's value and the average.

      Compress: This compresses the input array using Huffman coding, packing bits into the output array one byte at a time. It returns the total number of bytes written to the output.

2. **Measure**

Functions	$T_{\text{measured\_avg}}$(ns)	% of Total Latency	$T_{\text{measured\_avg}}$(cycles)
Scale	2.09032e+07	3.64	7.10708+e07
Filter horizontal	1.03589e+08	14.57	3.52202e+08
Filter vertical	1.04882e+08	16.39	3.56599e+08
Differentiate	2.59741e+07	3.64	8.83120e+07
Compress	3.22591e+08	61.94	1.09681e+09

   1) asdas

2) `&x[2].d[0]->b`

3)
```
#include <stdio.h>

int main(void) {
    int a[2][4] = { {10, 20, 30, 40}, {50, 60, 70, 80} };
    int *row0 = a[0];
    int *row1 = a[1];
    int *rows[2] = { row0, row1 };
    int **pp = rows;
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 4; ++j) {
            printf("pp[%d][%d] = %d\n", i, j, pp[i][j]);
        }
    }
    return 0;
}
```

4)
```
#include <stdio.h>
#include <stdlib.h>

void temp(int i) {
    int a[2];
    int b[3];
    int *c;
    int *d;

    c = (int *)malloc(sizeof(int) * 4);
    d = (int *)malloc(sizeof(int) * 5);

    printf("a (stack, 2 ints) : %p\n", (void*)a);
    printf("b (stack, 3 ints) : %p\n", (void*)b);
    printf("c (heap, 4 ints)  : %p\n", (void*)c);
    printf("d (heap, 5 ints)  : %p\n", (void*)d);

    free(c);
    free(d);

    return;
```

```
}

int main(void) {
    temp(0);
    return 0;
}
```

5) c[0] becomes 13
   invaild array indexing is detected and the program stops

6) char and unsigned char sums differ because unsigned char use all 8 bits for numbers but signed char has to use a bit for the sign, so they overflow at different sums

   char and unsigned char sums differ from their 'intsum' because intsum can use 32bits but chars can only use 8 so they overflow after 255.

7) Preprocessor: Handles stuff like include and define before compiling
   compiler: Translates the preprocessed C code into assembly or machine code
   Linker: Combines object files and libraries into executable

8) add the include path to the makefile -I
   copy the headerfile to local directory
   check name and path to make sure that they are correct

9) object file missing
   function not defined
   function signature does not match

3. **Debug an Application**

1) Done

2) Done

3) Done

4) 1) On line 5 I changed `"while (*s) s++;"` to `"while (*s++) l++;"`

2) Makefile:

```
release:
 gcc -Wall -o program program.c

debug:
 gcc -g -Wall -o program program.c
```

3) The secret message is: Well Done!!
4) while loop not executed at all, must be buggy len(). Steppting through len, l doesn't change at all. Look back at the code and noticed that s is incremented in while loop but l is not.

5) the -g flag of gcc adds debug symbols that maps addresses back to source files, and is read by gdb