Computer Lab Techniques

Name: Nicholas Hussain

Question:	1	2	3	4	5	6	7	Total
Points:	10	20	10	10	20	20	10	100
Score:								

1. (10 points) For the scrollbox, we created printable ASCII characters from space (' ', code 0x20) through tilde ('~', code 0x7E). Extended ASCII adds 128 additional characters, starting with Euro ('€', code 0x80) through y-umlaut ('ÿ', code 0xFF).

   Here is the file for the section sign, §, 0xA7. Fill in the Hex values.

```
/* Char_A7.h - Section sign
 *
 * Nicholas Hussain
 */

const byte Char_A7[10] = {
  0x00,      // ....|....
  0x1E,      // ...XXXX.
  0x20,      // ..X.|....
  0x3C,      // ..XXXX..
  0x22,      // ..X.|..X.
  0x1E,      // ...XXXX.
  0x02,      // ....|..X.
  0x3C,      // ..XXXX..
  0x00,      // ....|....
  0x00       // ....|....
};
```

2. (20 points) We want a Bash script which will take a list of filenames on the command line and print out how many lines are in each file.
For each of the following scripts, indicate whether it works. If it does not, show how to fix it by changing the *fewest* characters possible. (Note: output format is not important.)

Script A

☑ Works as is. ☐ Needs fixing.

```
#! /bin/bash

for file in $*
do
    wc -l $file
done
```

Script B

☐ Works as is. ☑ Needs fixing.

```
#! /bin/bash

for file in $@
do
    echo -n "$file " 
    grep -c "" $file
done
```

Script C

☐ Works as is. ☑ Needs fixing.

```
#! /bin/bash

for file in "$@"
do
    echo -n "$file "
    tr -dc '\n' < "file" | wc -l
done
```

Script D

☐ Works as is. ☑ Needs fixing.

```
#! /bin/bash

for file in $@
do
    echo -n "$file "
    awk 'END { print NR }' "$file"
done
```

3. (10 points) I wrote a program in C which crashed when it was run. The following text was copied and pasted directly from a shell window:

```
Program received signal SIGFPE, Arithmetic exception.
0x000000000040114f in zero_int (yr1=2010, yr2=2010, amt=100000)
    at mortgage.c:11
11          return amt / (yr2 - yr1);
(gdb) list
6     #include <stdio.h>
7     #include <stdlib.h>
8
9     double zero_int(int yr1, int yr2, int amt)
10    {
11        return amt / (yr2 - yr1);
12    }
13
14
15    /* the mortgage formula is from page 202 of
(gdb)
```

(a) What is the most likely reason the program stopped?
On line 2, yr1 = yr2. So it return amt/0. Which caused an error of divid by 0.

(b) What did the **user** (person running the program) most likely do wrong? How do you know? They tried to use zero int for the same year. Based on variable ~~name~~ names yr2 should be different year ~~or~~ at least 1 or more years after year 1.

(c) How should the program be changed to prevent this from happening again?
Have:

```
int yrdiff = yr2 - yr1
if (yrdiff <= 0) {
    echo "Year 2 is less than year 1";
}
else {
    return atm/yrdiff;
}
```

There is a simple check to see if there is a positive and greater than 0 yr difference

4. (10 points) The 'MPC' file format begins with the string My-Pic, terminated with a NULL character. Consider the following shell session:

```
elvix2:~/Rowan/Class/LabTech/MyPic 119> cat MPCsize.c
#include <stdio.h>
#include <string.h>

typedef struct {
    char           ident[7];
    unsigned short version;
    unsigned int   height;
    unsigned int   width;
} mypic_head;

int main(int argc, char *argv[])
{
    mypic_head  image_data;
    FILE       *image_file;

    image_file = fopen(argv[1], "r");

    if ( fread(&image_data, sizeof(mypic_head), 1, image_file) == 1 ) {
        printf("%s : version %d - %d x %d\n",
               argv[1],
               image_data.version, image_data.width, image_data.height);
    } else {
        printf("Problem reading %s\n", argv[1]);
    }

    fclose(image_file);
}
elvix2:~/Rowan/Class/LabTech/MyPic 120> ./MPCsize IMG-1021.mpc
IMG-1021.mpc : version 9 - 209 x 21
elvix2:~/Rowan/Class/LabTech/MyPic 122>
```

Fill in the grid with the hexadecimal values of the bytes in the header of **IMG-1021.mpc**, one byte per cell. If a value is unknowable, put '?'. For the first byte past the header, write 'END' in the box. (For character data, you can write the character instead of the ASCII value in hex.)

The grid is 8 blocks wide; the columns have been numbered, and the rows are marked with their start values. Byte 0, the first byte in the file, goes in the upper-left. Byte 1, the second byte in the file, goes in Row 0x00, Column 1. Byte 8 goes in Row 0x08, Column 0. Byte 17 goes in Row 0x10, Column 1. (Because 0x10 is 16, plus 1 is 17.)

	0	1	2	3	4	5	6	7
0x00	M	y	2D	50	69	63	00	?
0x08	?	?	?	?	?	?	?	?
0x10	?	09	01	00	15	?	?	?
0x18	?	?	?	?	?	?	?	END

(The first two have been done for you.)

Hint: remember the discussion of alignment and bus errors from the debugging class.

5. (20 points) Here are four C programs that count the bits on in an integer; that is, you run './cb 5' and it will print '2' (because 5 in binary is 101, and two bits are turned on). For each program, indicate whether it works. If it does not, show how to fix it by changing the *fewest* characters possible. Note that these programs will crash if not given an argument; that's not considered an error per the design spec.

Program A - Add braces to if/for sum;

☑ Works as is. ☐ Needs fixing.

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

int main(int argc, char *argv[])
{
    // warning: assumes argv[1]!
    int val = atoi(argv[1]);
    int sum = 0;

    for (int i = 0; i < 32; i++){
        if (val & (1 << i)){
            sum += 1;
        }
    }
    printf("%d\n", sum);
}
```

Program B

☐ Works as is. ☑ Needs fixing.

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

int main(int argc, char *argv[])
{
    // warning: assumes argv[1]!
    int val = atoi(argv[1]);
    int sum = 0;

    for (int i = 31; i >= 0; i--){
        if (val | 1 << i)
            sum += 1;
    }
    printf("%d\n", sum);
}
```

Program C

☐ Works as is. ☑ Needs fixing.

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

int main(int argc, char *argv[])
{
    // warning: assumes argv[1]!
    int val = atoi(argv[1]);
    int sum = 0;

    while ( val ) {
        if ( val < 0 )
            sum ++;
        val <<= 1;
    }

    printf("%d\n", sum);
}
```

Program D

☐ Works as is. ☑ Needs fixing.

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

int main(int argc, char *argv[])
{
    // warning: assumes argv[1]!
    int val = strtoul(argv[1], NULL, 10);
    int sum = 0;

    while (val) {
        sum += val & 1;
        val >>= 1;
    }

    printf("%d\n", sum);
}
```

6. (20 points) Here are four C programs to translate binary to decimal; that is, you run './pb 101' and it will print '5'. For each program, indicate whether it works. If it does not, show how to fix it by changing the *fewest* characters possible. Note that these programs will crash if not given an argument; that's not considered an error per the design spec.

Program A

☐ Works as is. ☑ Needs fixing.

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

int main(int argc, char *argv[])
{
    int val = 0;
    // warning: assumes argv[1]!
    int len = strlen(argv[1]);

    for (int i = 0; i < len; i++) {
        val >>= 1;
        if ( argv[1][i] == '1' )
            val |= 1;
    }

    printf("%d\n", val);
}
```

Program B

☑ Works as is. ☐ Needs fixing.

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

int main(int argc, char *argv[])
{
    int val = 0;
    // warning: assumes argv[1]!
    int len = strlen(argv[1]);

    for (int i = 0; i < len; i++) {
        val = val * 2 +
              ( argv[1][i] - '0');
    }

    printf("%d\n", val);
}
```

Program C

☑ Works as is. ☐ Needs fixing.

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

int main(int argc, char *argv[])
{
    int val = 0;
    // warning: assumes argv[1]!
    int len = strlen(argv[1]) - 1;

    for (int i = len; i >= 0 ; i--) {
        if ( argv[1][i] == '1' )
            val |= 1 << len - i;
    }

    printf("%d\n", val);
}
```

Program D

☐ Works as is. ☑ Needs fixing.

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

int main(int argc, char *argv[])
{
    int val = 0;
    // warning: assumes argv[1]!
    int len = strlen(argv[1]) - 1;

    for (int i = len; i > 0 ; i--) {
        val += ( argv[1][i] - '0' )
               << (len - i);
    }

    printf("%d\n", len);
}
```

7. (10 points) Consider the following Makefile:

```
zap : perm
        ./zip > zap

zip: zip.start
        (cat zip.start ; echo 'echo zop') > zip

perm : zip
        chmod +x zip

zip.start:
        echo '#! /bin/bash' > zip.start
```

(a) If this Makefile was in an otherwise-empty directory named **m3**, what commands would be executed, *in what order*, when the user ran 'make'?

1) echo '#! /bash ... zip.start
2) (cat zip.start ; ... > zip
3) chmod +x zip
4) ./zip > zap

(b) Draw the subtree, showing all the files in **m3** after 'make' had finished.

m3/
- Makefile
- perm
- zap
- zip ← this is an executable
- zip.start

Extra Credit For Helping Future Students *(4 points)*:

A) Juvpu nffvtazrag jnf gur zbfg rqhpngvbany/vagrerfgvat?
The clock was most interesting because it can be used and put onto an arduino and used for a real clock.

B) Juvpu nffvtazrag jnf gur yrnfg rqhpngvbany/vagrerfgvat?
the image/gif size one. Its easy to find that info if you have access to that file

C) Jung fubhyq V fcraq zber gvzr ba?
Spend more time on bash scripting because that seems to have more importance than find image size.

D) Qhzo wbxrf, tnzrf, naq bgure fvyyl fghss uryc xrrc gur pynff sebz orvat n grqvbhf vasb qhzc. Anzr bar V fubhyq qb ntnva arkg frzrfgre.
Try and remove 3:F/imax lab. Move Clock ur. And create a simple game students can build. We have access to key binds so it seems it can work.