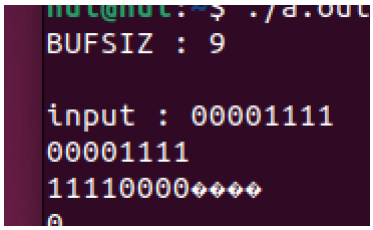
1.
소속 : 소프트웨어학부
학번 : 20210292
이름 : 김희영
개발환경 : 맥 -> 리눅스 터미널

2. 수정 및 작성한 소스코드에 대한 설명
(1) 20210292.c 파일

```
#include <stdio.h>
#include <malloc.h> // 메모리 동적할당을 위한 헤더 파일
#include “20210292.h” // .h파일
long filesize;
int main(void) {
	FILE *input = fopen("input.txt", “r"); // fopen을 활용하여 input.txt을 읽어온다.
	if (input==NULL) { // input.txt 파일이 NULL일 경우 “not file”을 출력하고 리턴한다.
		printf("not file");
		return 0;
	}

	// 파일 사이즈를 찾기 위한 과정이다.
	fseek(input, 0, SEEK_END); // fseek를 통해 파일의 끝 지점을 이동한다.
	filesize = ftell(input); // ftell을 통해 파일 사이즈를 읽어온 후 filesize 변수 선언한다.
	printf("BUFSIZ : %ld\n”,filesize);
	rewind(input); // 다시 파일의 맨 처음 위치로 이동한다.

	// 메모리 공간을 동적 할당한다.
	char *buffer = (char*) malloc(filesize*sizeof(char));
	fgets(buffer, filesize, input); // fgets을 통해 buffer안에 input을 넣는다.
	printf("input : %s\n”,buffer);

	// 다양한 데이터 유형으로 출력하기 위한 함수들
	toSignedChar(buffer); // signed char 형태로 출력하는 함수이다.
	toAscii(buffer); // ASCII code 형태로 출력하는 함수이다.
	toUnsignedChar(buffer); // unsigned char  형태로 출력하는 함수이다.
	toSignedInt(buffer); // signed int 형태로 출력하는 함수이다.
	toUnsignedInt(buffer); // unsigned int  형태로 출력하는 함수이다.
	toFloat(buffer); // float  형태로 출력하는 함수이다.
	toDouble(buffer); // double  형태로 출력하는 함수이다.

	free(buffer); // 사용이 끝났으므로 free함수를 통해 메모리를 해제한다.
	return 0;
}

// char형태로 저장되어있는 buffer를 decimal number형태로 변환하여 리턴해준다.
long binaryToDecimal(const char* buffer, int i, int b) { // 버퍼와 for문에서 몇번째인지를 나타내는 i와 bit 수를 파라미터로 받는다.
	char data[b]; // 데이터마다 몇 비트씩 읽어야 하는지 다르기 때문에 b를 통해 배열의 크기는 정한다.
	for (int j = 0; j < b; ++j) {
		data[j] = buffer[i * b + j]; // 리틀엔디안이므로 뒤에서부터 읽어와야 하기 때문에 data 배열에 해당 데이터의 비트만큼만 저장한다.
```

```
        }
        long a = 0; // 십진수를 담아 리턴할 변수
        for (int j=0; j<b; j++) {
                a = (a << 1) + (data[j] - '0'); // (data[j]-'0') 를 통해서 비트를 하나 읽어와  '0'의 ascii code를 빼주어 십진수로 변환한다. 그 이후 비트연산자(<<)를 통해서 a를 왼쪽으로 비트 1만큼 shift 한 이후 (data[j]-'0')을 더해서 a에 저장한다. a<<1만큼 shift할 경우 a에 2를 곱한 것을 의미한다.
        }
        return a; // 십진수로 변환한 것을 리턴한다.
}

void toSignedChar(const char* buffer) { // signed char 로 변환하여 출력하는 함수
        printf("1. signed char : ");
         for (int i =(filesize-1)/8-1; i>=0 ; i--) { // filesize는 \0까지 포함하기 때문에 -1을 한 후 1바이트이므로 8로 나눈 후 -1을 한다. for문을 통해 signed char가 반복하여 출력하는 형식으로 하였다.
                 printf("%d ", (signed char)binaryToDecimal(buffer, i, 8); //char형태는 1바이트, 즉 8비트이므로 b는 8로 넘긴다.
        }
        printf("\n"); // 끝났을 경우 줄 바꿈을 해준다.
}
void toAscii(const char* buffer) { // ascii code 로 변환하여 출력하는 함수
        printf("2. ASCII codes : ");
          for (int i= (filesize-1)/8-1; i>=0; i--) { // filesize는 \0까지 포함하기 때문에 -1을 한 후 1바이트이므로 8로 나눈 후 -1을 한다. for문을 통해 ascii code가 반복하여 출력하는 형식으로 하였다.
                int a = binaryToDecimal(buffer, i, 8); // 십진수 a를 반환한다.
                if (0 <= a && a <= 126) {
                        printf("%c ", a); // ascii code 범위인 0부터 126까지만 출력한다.
                }
                else {
                        printf(". "); // 과제 조건에 의해 범위가 아닐 경우 . 을 출력한다.
                }
        }
        printf("\n"); // 끝났을 경우 줄 바꿈을 해준다.

}
```

3. 과제 수행 중 발생한 문제점과 해결 방법
(1)리틀엔디안으로 출력하려는 도중에 이상한 문자가 같이 출력되는 오류가 발생하였다.



(정확히 나오는지 확인하기 위해서 출력한 것입니다.)

-> 출력 형식에 의해 발생한 오류이기 때문에 수정하였다.

(2) segmentation fault (core dumped) 오류 발생하였다.

```
BUFSIZ : 9
input : 11110000
Segmentation fault (core dumped)
```

-> Segmentation fault는 프로그램이 메모리를 잘못 액세스하여 발생하는 오류이다. 배열 범위를 벗어난 인덱스를 접근하여 발생한 오류이기 때문에 배열을 수정하였다.

(3) assignment of read-only location 에러

```
hui@hui:~$ gcc 20210292.c
20210292.c: In function ‘charToBinary’:
20210292.c:47:31: error: assignment of read-only location ‘*(result + (sizetype)
((long unsigned int)size++ * 4))’
   47 |                 result[size++]=binaryToDecimal(data);
      |                               ^
```

-> 읽기 전용 메모리에 값을 수정하려고 하여 발생한 에러이다. result가 읽기 전용 메모리 위치를 가리키는 포인터라는 의미이다. 다 const int* result라고 작성하여 발생한 에러이기 때문에 수정을 해야 하는 부분에는 const를 지워서 작성하였다.
-> result 부분이 없는 방향으로 소스 코드를 수정하였다.

(4) lvalue required as increment operand

```
20210292.c: In function ‘toSignedChar’:
20210292.c:61:29: error: lvalue required as increment operand
   61 |                         data++;
      |                             ^~
```

-> 직접적으로 ++을 통해 위치를 변경하지 않고 for문을 통해 위치를 변경하도록 하였다. data[j] 와 같은 식으로 변경하였다.

(5) bus error (core dumped)

```
hui@hui:~$ ./a.out
BUFSIZ : 65
input : 00101000101100010100010110001010101010101010111100001100011110000
1. signed char : 112 12 -81 -86 -118 69 -79 40
2. ASCII codes : p
                 . . . E . (
3. unsigned char : 112d 12d 175d 170d 138d 69d 177d 40d
Bus error (core dumped)
```

-> 프로그램이 잘못된 메모리를 접근하거나 잘못된 방식으로 메모리에 접근하여 발생하는 오류이다. bus error가 발생할 수 있는 원인들 중에서 해당 되는 이유로는 잘못된 포인터 사용과 메모리 할당 문제, 쓰기 보호된 메모리에 쓰기를 시도하거나 메모리 정렬 조건을 만족하지 않으면 발생할 수 있다. 발생한 이유로 추측 가능한 것이 메모리를 초과한 것이다. 따라서 원래는 함수에 직접 코드를 작성하여 수행하는 방식이였는데 메모리를 초과한 것으로 판단하여 binaryToDecimal 함수를 따로 빼서 진행하는 식으로 소스 코드를 작성하였다.

(6) #include <stdio.h> : did you forget to

```
hui@hui:~$ make
gcc -c -o 20210292.o 20210292.c -I.
20210292.c: In function ‘main’:
20210292.c:4:9: error: unknown type name ‘FILE’
    4 |         FILE *input = fopen("input.txt", "r");
      |         ^~~~
20210292.c:2:1: note: ‘FILE’ is defined in header ‘<stdio.h>’; did you forget to
 ‘#include <stdio.h>’?
    1 | #include "20210292.h"
  +++ |+#include <stdio.h>
    2 | long filesize = 0;
20210292.c:4:23: warning: implicit declaration of function ‘fopen’ [-Wimplicit-f
unction-declaration]
    4 |         FILE *input = fopen("input.txt", "r");
      |                       ^~~~~
20210292.c:4:23: note: ‘fopen’ is defined in header ‘<stdio.h>’; did you forget
to ‘#include <stdio.h>’?
```

-> 20210292.h 파일 안에 #include <stdio.h> 등 작성하여 발생한 오류이다. 표준 입출력 함수를 사용하기 위한 라이브러리이기 때문에 .c파일에 선언하였다.

(7) signed float , signed double

함수를 통해 십진수를 받아 (float)함수리턴값과 같은 형식으로 진행하였는데 계속 unsigned 형식으로 출력하여 결과값이 틀리는 현상이 발생하였다. 따라서 출력형식을 변환하여 출력하려 하였지만 출력형식은 무조건 %.4f로 해야했기 때문에 %+.4f로 바꾸었다. 하지만 이 방식 또한 unsigned값에 +가 붙는 형식으로 음수 부분이 출력되지 않았다. 따라서 먼저 signed int형으로 변환하여 해당 값에 다시 (float)를 붙이는 방식으로 하였다. double에 경우는 signed long형으로 변환한 후 해당 값에 (double)을 붙이는 방식으로 하였다.

4. 실행 결과

```
hui@hui:~$ make
make: '20210292' is up to date.
hui@hui:~$ ./20210292
BUFSIZ : 65
input : 00101000101100010100010110001010101010101010111100001100011100000
1. signed char : 112 12 -81 -86 -118 69 -79 40
2. ASCII codes : p
                 . . . E . (
3. unsigned char : 112 12 175 170 138 69 177 40
4. signed int : -1431368592 682706314
5. unsigned int : 2863598704 682706314
6. float : -1431368576.0000 682706304.0000
7. double : 29322012942663055360.0000
hui@hui:~$
```