unsigned int를 선언하고 사용하는 방법:
"unsighed int 변수명"을 통해 unsigned int를 선언할 수 있다.
unsigned int는 그냥 int값처럼 사용이 가능하나, 범위 값이 다르고 특별히 %d로 표현하지 않고 주로 %u로 표현한다는 점을 유의해야 한다. unsigned int를 %d로 처리하면 범위, 부호 오류가 날 수 있으니 조심해야 한다.

unsigned int를 사용하는 목적:
범위값만을 보면 음수값을 원치 않고 int 자료형의 메모리 값을 양수영역으로 몰아서 사용하고 싶을 때 쓴다.
쉬프트 연산자 <<를 사용하면 비트값이 왼쪽으로 한칸 이동하는데 이것을 전체 값에 2배를 해준 것이라고 볼수도 있다. 여기서 중요한 점은 int 자료형은 맨 앞의 비트가 정수값의 부호를 결정하기 때문에 int에 <<연산자를 적용하면 그 값이 항상 2배가 된다는 보장은 없다. 따라서 unsigned int를 사용하여 부호에 대한 비트를 그냥 정수를 표현하는 비트로 바꾸어 <<연산자를 적용하면 그 값이 2배가 되도록 만들 수 있다. (물론 unsigned int의 범위 내에서만 성립한다.)

프로그램의 설계 과정:
1. 주어진 문자열의 숫자만큼 문자열을 받는 함수 inputname을 만들었다.
2. unsigned int의 범위, 자릿수를 생각해서 10자리 문자열을 함수 inputname로 받고, 이 숫자로만 구성되어있을 때 10자리 문자열을 unsighed int의 형태로 변환하여 반환하는 함수 getunsignedint를 만들었다.
(여기서 아쉽게도 atoi함수는 사용하지 못했다. 왜냐하면 atoi함수는 문자열을 int값으로 변환해버리고 int범위를 초과하는 unsigned int값에 대해서는 INT_MAX를 반환해 버리기 때문이다.)
3. unsigned int 값을 입력받아 비트값을 거꾸로 뒤집어서 새로운 unsigned int 값을 반환하는 함수 reversBits를 만들었다.
4. unsigned int 값을 입력받아 비트값을 출력하는 함수 displayBits를 만들었다.
5. getunsignedint함수로 unsigned int 값을 받아 displayBits로 그 unsigned int의 비트값을 출력하고, reversBits로 비트를 뒤집어서 displayBits로 뒤집어진 unsigned int의 비트값을 출력한다.

프로그램의 작동 원리:
1. inputname 함수:
포인터 배열과 배열의 갯수를 입력해주면, for문으로 배열의 갯수만큼 getchar하는 데 ₩n이 나타나면 break해버린다. 끝까지₩n이 안나타나면 버퍼를 지우고 다시 for문을 돌린다. 만약 위 조건을 만족하는 배열이 나오면 ₩n자리를 NULL로 바꾸어준다.
2. getunsignedint 함수:

unsigned int의 최댓값의 자릿수는 10자리이다. 그래서 inputname 함수를 이용해 11자리 배열을 받는다. (NULL값을 포함하니까) NULL문자 이전의 모든 배열이 0~9사이에 있는지 확인한다. 만일 적어도 하나의 배열이 0~9사이에 없다면 다시 inputname 함수를 이용해 11자리 배열을 받는다. NULL문자 이전의 모든 배열이 0~9사이에 있다면 그 배열을 long자료형에 정수형으로 변환한다. 그 정수가 unsigned int의 범위 안에 있다면 그 값을 반환하고 없다면 처음부터 앞선과정을 반복한다.

3. reversBits함수:

어떤 unsigned int를 인풋으로 받는다. unsigned int reverse = 0로 새로운 unsigned int변수를 선언, 초기화한다. unsigned int mask = 1으로 마스크를 생성한다. for문으로 32번 다음과 같은 과정을 반복한다.

i) 우선 reverse를 2배해준다. 이는 비트값을 한칸씩 왼쪽으로 옮기는 것과 같다.

ii) 만일 인풋으로 주어진 unsigned int과 마스크의 값을 &연산자로 계산했을때 마스크와 일치하면 reverse에 1을 더해준다.

iii) 마스크를 <<연산자로 왼쪽으로 한칸 움직여준다.

for문 이후 reverse값을 반환한다.

4. displayBits 함수:

어떤 unsigned int를 인풋으로 받는다. unsigned int mask = 1 << 31로 마스크 값을 생성한다.

받은 인풋값을 먼저 printf로 보여준다.

for문으로 32번 다음과 같은 과정을 반복한다.

i) 만일 인풋으로 주어진 unsigned int과 마스크의 값을 &연산자로 계산했을때 마스크와 일치하면 '1'을 출력하고, 아니면 '0'을 출력한다.

ii) 8의 배수번째 시도에는 띄어쓰기를 출력한다.

iii) 마스크를 >>연산자로 오른쪽으로 한칸 움직인다.

5. main 함수:

unsigned int input을 선언한다. getunsignedint함수에 unsigned int의 최대값, 최솟값을 넣어주고, getunsignedint함수의 리턴값을 input에 넣어준다.

printf로 "Before bits are reversed:"라고 출력하고 input의 비트를 displayBits(input)로 출력한다 printf로 "After bits are reversed:"라고 출력하고 뒤집어진 input의 비트를 출력하기 위해displayBits 함수와 reversBits 함수를 이용한다. displayBits(reversBits(input))와 같이 코딩할 수 있다. 0을 리턴하고 프로그램을 종료한다.

어려웠던 점:

연산우선순위를 생각하지 못했다. 처음에 number&mask == mask라고 적어서 먼저

mask == mask가 연산되고 그 다음에 number&이 연산되어서 원하지 않는 결과가 나왔다.

(number&mask) == mask로 고친 후에 정상적으로 작동함을 알게 되었다.
앞으로는 연산 우선순위도 항상 고려하고 연산자가 많아지면 ()를 사용하는 습관을 가지자.