1A)
```
while (x) {
      val ^= x;
      x >>= 1;
}
return val & 1;
```

1B) returns 1 if there are odd number of bits that are 1, 0 otherwise

2A)
```
for(i=0; i<32; i++){
      val = (val<<1) |(x & 1);
      x >>= 1;
}
```

2B) This function reverses the bits in the paramentter passed in. The most significant bit becomes the least. And the least significant bit becomes the most

3A) operator is division (/)

3B)
```
leal 3(%edx), %eax      ; temp = x+3
testl %edx, %edx        ; temp & temp
cmovns %edx, %eax       ; if result sign is +; temp = x
sarl $2, %eax           ; return temp >> 2 (temp/4)
```

4A) 0x80003C (%ebp-$4)

4B) 0x800014 (%esp-$40)

4C) x → -4(%ebp) = 0x800038

y → -8(%ebp) = 0x800034

4D)

0x80003C	0x800060
0x800038	0x53
0x800034	0x46
...	...
0x80001C	0x800038
0x800018	0x800034
0x800014	0x300070

4E) Address 0x800020 to 0x800033 are unused

5A) Holds the parameter x

5B)

```
if(!x) {
    return 0;
}
unsigned nx = x>>1;
int rv = rfun(nx);
return (x&1)+rv;
```

5C) The function sums the number of bits that are 1

6) M=5 N=7