question 1

1. 0
2. 1
3. 1
4. 0
5. 1
6. 0
7. 1
8. 0
9. 1
10. 1

question 2

1.
```
$ gcc -m32 lab2_b.c -o main
$ ./main
5
```

2.
```
$ objdump -d main | grep -A 14 "<max>:"
0000051d <max>:
 51d:	55                   	push   %ebp
 51e:	89 e5                	mov    %esp,%ebp
 520:	e8 5f 00 00 00       	call   584 <__x86.get_pc_thunk.ax>
 525:	05 b3 1a 00 00       	add    $0x1ab3,%eax
 52a:	8b 45 08             	mov    0x8(%ebp),%eax
 52d:	3b 45 0c             	cmp    0xc(%ebp),%eax
 530:	7e 05                	jle    537 <max+0x1a>
 532:	8b 45 08             	mov    0x8(%ebp),%eax
 535:	eb 03                	jmp    53a <max+0x1d>
 537:	8b 45 0c             	mov    0xc(%ebp),%eax
 53a:	5d                   	pop    %ebp
 53b:	c3                   	ret

0000053c <main>:
```

3. 在 0x000535 处有一跳转指令

```
 535:	eb 03                	jmp    53a <max+0x1d>
```

使用了两个字节来表示该指令，第一个字节 `eb` 表示 `jmp` ，第二个字节 `03` 表示目标指令的地址与紧跟在跳转指令后面那条指令的地址之间的差，所以目标指令的地址为 537 + 3 = 53a 处

question 3

```c
int fun_a(unsigned x) {
    int val = 0;
    while (x) {
        val ^= x;
        x >>= 1;
    }
    return val & 1;
}
```

返回 x 所有位的异或值

question 4

```c
int loop(int x, int n) {
    int result = 0;
    int mask;
    for (mask = 1; mask != 0; mask = mask << (char)n) {
        mask |= x & mask;
    }
    return result;
}
```

question 5

1. 条件传送会对 then-expr 和 else-expr 都求值，而 `*xp` 在 `p == NULL` 时是错误的，当测试为假时， `cmovne` 指令对 `xp` 的间接引用仍然发生了，导致了一个间接引用空指针的错误。

2.
   ```c
   int cread(int *xp) {
       int tmp = 0;
       return *(xp ? xp : &tmp);
   }
   ```

question 6

```c
int main() {
    int a = 8, b = 3, prod;
    asm (
        "imul %1, %2\n\t"
        "movl %2, %0"
        : "=r" (prod)
        : "r" (a), "r" (b)
        :
    );
    printf("%d\n", prod);
    return 0;
}
```

output:

```
$ gcc -m32 lab2_b.c -o main
$ ./main
24
```