question 1

```
int loop(int x, int y, int n) {
    int result = 0;
    int i;
    for (i = n - 1; i >= 0; i = i - x) {
        result += x * y;
    }
    return result;
}
```

两条缺失的跳转指令分别是

```
jl    .L4
jge   .L6
```

question 2

```
int decode2(int x, int y, int z) {
    y -= z;
    x *= y;
    int res = y;
    res <<= 31;
    res >>= 31;
    res ^= x;
    return res;
}
```

question 3

```
unsigned replace_byte (unsigned x, int i, unsigned char b) {
    return x & ~(0xFF << (i * 8)) | (b << (i * 8));
}
int main() {
    int x = 0x12345678;
    printf("%x\n", replace_byte(x, 0, 0xab));
    printf("%x\n", replace_byte(x, 3, 0xab));
}
```

output:

```
$ gcc -m32 main.c
$ ./a.out
123456ab
ab345678
```

question 4

1. K = 31

   $31 = 2^5 - 2^0$

   ```
   int mul_31(int x) {
       return (x << 5) - x;
   }
   ```

2. K = -3

   $-3 = -(2^1 + 2^0)$

   ```
   int mul_-3(int x) {
       return -((x << 1) + x);
   }
   ```

3. K = 1032

   $1032 = 2^{10} + 2^3$

   ```
   int mul_1032(int x) {
       return (x << 10) + (x << 3);
   }
   ```

4. K = -48

   $-48 = -(2^5 + 2^4)$

   ```
   int mul_-48(int x) {
       return -((x << 5) + (x << 4));
   }
   ```

question 5

```
typedef unsigned float_bits;
/* Compute |f|. If f is NaN, then return f. */
float_bits float_absval(float_bits f) {
    int exp = (f & (0xff << 23)) >> 23;
    int frac = f & 0x7fffff;
    if (exp == 0xff && frac) return f;
    else return f & ~(1 << 31);
}
```

为了方便测试，定义了一个 inf-float 相互转化的 union:

```c
typedef union {
    float_bits i;
    float f;
}int_f;
int main() {
    int_f t, abst;
    while (1) {
        printf("input: ");
        scanf("%f", &t.f);
        abst.i = float_absval(t.i);
        printf("f   %f\n|f| %f\n\n", t.f, abst.f);
    }
}
```

测试:

```
$ ./a.out
input: -inf
f   -inf
|f| inf

input: -1234
f   -1234.000000
|f| 1234.000000

input: nan
f   nan
|f| nan

input: -nan
f   -nan
|f| -nan
```