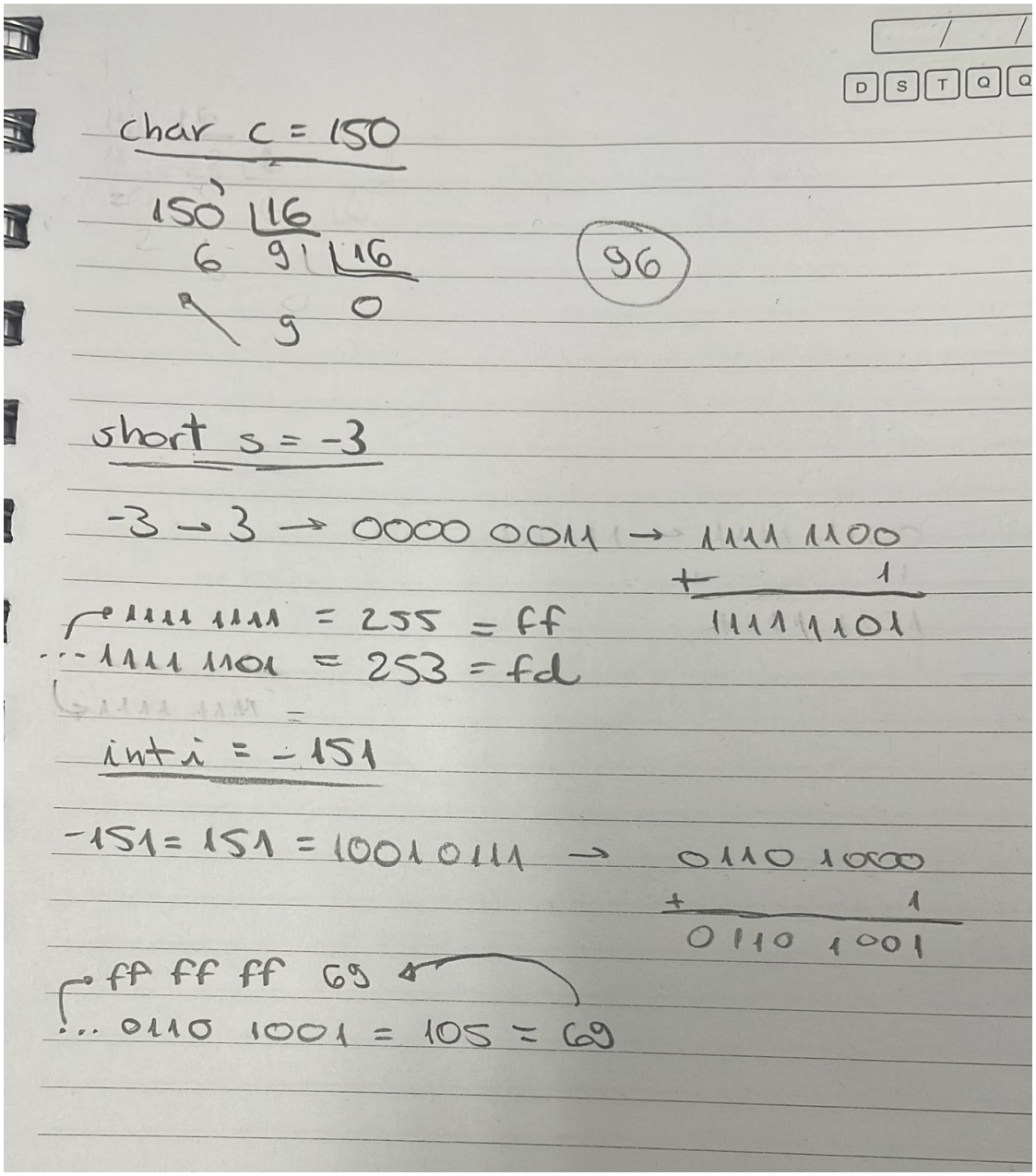
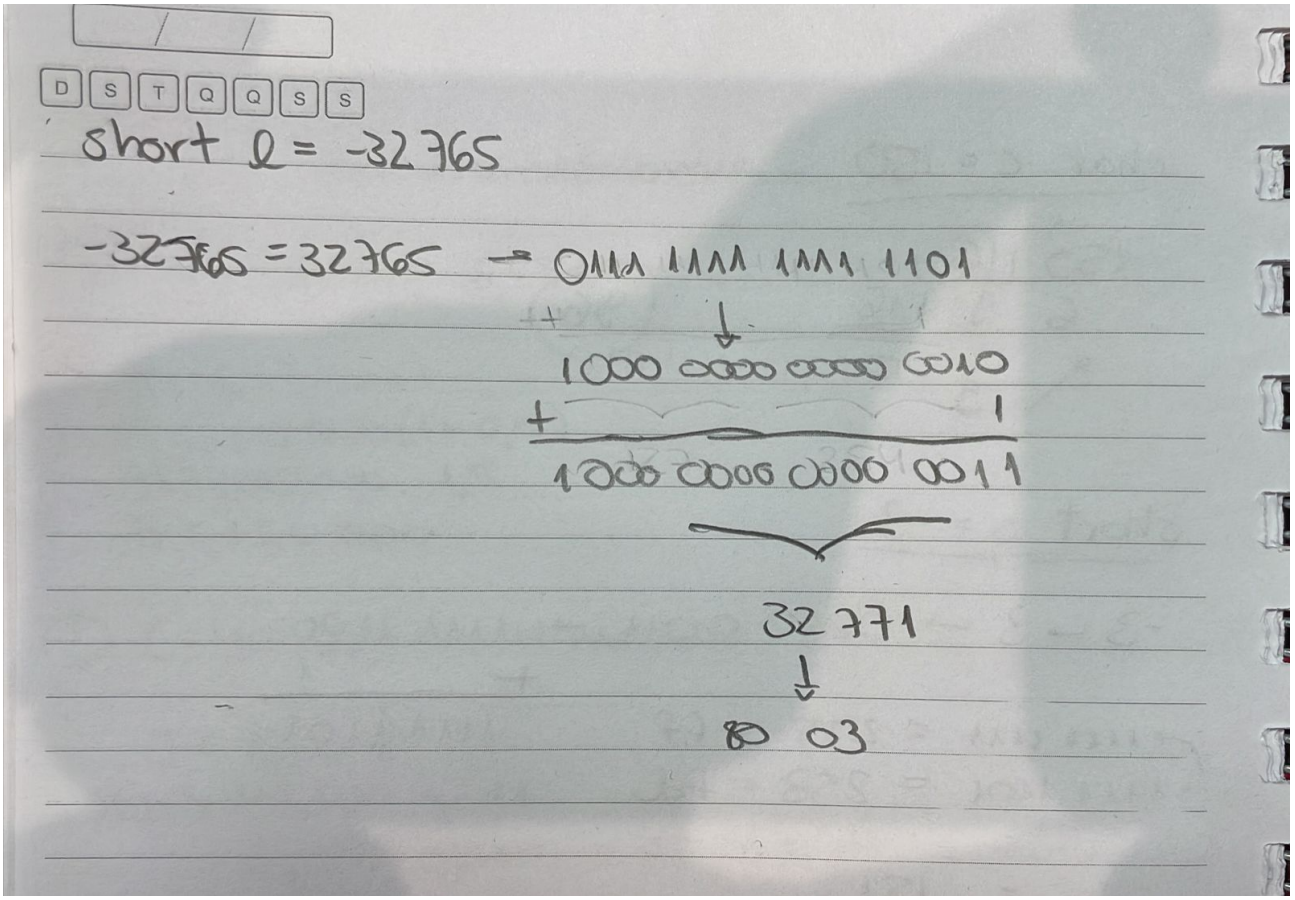
Nome: Lucas Lemos
Matrícula: 2110013

1ª questão




```
dump de c:
0x7ffd4b5523c1 - 96
dump de s:
0x7ffd4b5523c2 - fd
0x7ffd4b5523c3 - ff
dump de i:
0x7ffd4b5523c4 - 69
0x7ffd4b5523c5 - ff
0x7ffd4b5523c6 - ff
0x7ffd4b5523c7 - ff
```

2ª questão




```
l=-32765, k=32771
dump de l:
0x7ffecd7d5a74 - 03
0x7ffecd7d5a75 - 80
dump de k:
0x7ffecd7d5a76 - 03
0x7ffecd7d5a77 - 80
```

O número -32765, quando calculado o seu complemento a 2, encontra-se o valor binário para o número unsigned 32771. Por esse motivo, o *dump* de ambos valores é igual.

3ª questão

```
int xbyte (packed_t word, int bytenum) {
    unsigned int x = word;
    printf ("%x \n",x);
    x = x>>(bytenum*8);
    x = x & 0x00000FF;
    char auxiliar = x;
    return auxiliar;
}
```

Inicialmente, inicializei a variável x como *unsigned int*, representando a estrutura que empacota os 4 bytes. Depois, utilizei as operações e bits para isolar o byte desejado e capturá-lo em seguida. Por fim, criei uma variável tipo *char*, para que o valor seja truncado e em seguida estendido novamente, mantendo o bit mais significativo, identificador do sinal.

4ª questão

A diferença entre o programa 1 e 2 ocorre devido ao tipo que as variáveis foram inicializadas em cada caso. No programa 1, o valor hexadecimal 0xFFFFFF representa o decimal -1, já que foi inicializada como signed int. Nesse caso -1<2 e o programa responde que sim.
Por outro lado, no programa 2, as variáveis foram inicializadas com o tipo unsigned int, então o valor decimal corresponde a 4294967295. Por esse motivo, o programa responde que não.
O programa 3 tem esse comportamento, já que, em expressões com int do tipo com e sem sinal, todos são tratados como unsigned, por isso o valor 0xFFFFFF, não representa o número -1, mas, sim, 4294967295.

5ª questão

O código gerado deve adicionar 24 cópias do bit mais significativo à esquerda, devido à extensão com sinal. Desse modo, o unsigned int ficará 0xFFFFFFFF.