Nome: Thiago Pereira Camerato
matrícula: 2212580

Programa 1:
Variável `c`:
A variável `c` é do tipo `char` e recebe o valor 150. Em representação de complemento de 2, esse valor é 0x96 em hexadecimal, que é positivo. Portanto, o byte de `c` será impresso como `0x96`.
Variável `s`:
A variável `s` é do tipo `short` e recebe o valor -3. Em representação de complemento de 2, esse valor é 0xFFFD em hexadecimal. O byte menos significativo será `0xFD`, e o byte mais significativo será `0xFF`.
Variável `i`:
A variável `i` é do tipo `int` e recebe o valor -151. Em representação de complemento de 2, esse valor é 0xFFFFFF69 em hexadecimal. Os bytes serão impressos na seguinte ordem: `0x69`, `0xFF`, `0xFF`, `0xFF`.
Programa 2:
Variável `l`:
A variável `l` é do tipo `short` e recebe o valor -32765. Em representação de complemento de 2, esse valor é 0x8007 em hexadecimal. O byte menos significativo será `0x07`, e o byte mais significativo será `0x80`.
Variável `k`:
A variável `k` é do tipo `unsigned short` e recebe o valor 32771. Em hexadecimal, esse valor é 0x8003. O byte menos significativo será `0x03`, e o byte mais significativo será `0x80`.
Agora, vamos executar o programa para confirmar os resultados.

dump de c:
0x7fff5cc96ff7 - 96
dump de s:
0x7fff5cc96ff6 - fd
0x7fff5cc96ff5 - ff
dump de i:
0x7fff5cc96ff4 - 69
0x7fff5cc96ff3 - ff
0x7fff5cc96ff2 - ff
0x7fff5cc96ff1 – ff

Programa 2:
l=-32765, k=32771
dump de l:
0x7fff6ba63b12 - 07
0x7fff6ba63b11 - 80
dump de k:
0x7fff6ba63b0e - 03
0x7fff6ba63b0f - 80
Os resultados impressos pelos programas coincidem com as análises feitas anteriormente. Cada byte das variáveis foi impresso em sua representação hexadecimal correta.
Implementação da Função `xbyte`:
A função `xbyte` deve extrair o byte indicado (de 0 a 3) a partir de um valor inteiro `packed_t` e retornar esse byte como um inteiro com sinal de 32 bits.
Explicação:

- `unsigned char *p = (unsigned char *)&word;` cria um ponteiro para o início da representação em bytes de `word`.
- `(char)p[bytenum]` converte o byte para `char` (considerando que `char` tem 8 bits e é tratado como sinalizado).
- `(int)` então converte o `char` para `int` mantendo o sinal.

Agora, vamos testar a função `xbyte` com os exemplos fornecidos.
Exemplos de Uso:
./prog3 01abcd02 0
Resultado esperado: `00000002 2`
./prog3 11a032b5 0
Resultado esperado: `fffffb5 -75`
./prog3 11a032b5 1
Resultado esperado: `00000032 50`
./prog3 11a032b5 2
Resultado esperado: `fffffa0 -96`
./prog3 11a032b5 3
Resultado esperado: `00000011 17`
./prog3 abcd 3
Resultado esperado: `00000000 0`
./prog3 abcd 1
Resultado esperado: `fffffab -85`
./prog3 zzzz 1
Resultado esperado: `pane: numero invalido!`

Programa 4:
1)
Neste programa, `x` é inicializado com o valor máximo representável em um `int` de 32 bits (`0xffffffff` em hexadecimal) e `y` é inicializado com o valor 2. Depois, são impressos os valores de `x` e `y`. Em seguida, verifica-se se `x` é menor do que `y`. Como `x` é um número negativo e `y` é um número positivo, a condição `(x<y)` é falsa. Portanto, será impresso "x é menor do que y? nao".
2)
Neste programa, tanto `x` quanto `y` são declarados como `unsigned int`. Portanto, `x` recebe o valor máximo representável em um `unsigned int` (que também é `0xffffffff` em hexadecimal). O restante do programa é semelhante ao Programa 1.
3)
Neste programa, `x` é declarado como `int` e inicializado com o valor máximo representável em um `int`, e `y` é declarado como `unsigned int` e inicializado com o valor 2. O restante do programa é semelhante aos programas anteriores.

Programa 5:
signed char sc = -1;
unsigned int ui = sc;
Ao atribuir um valor negativo a `sc`, o código gerado precisa converter o valor para um `unsigned int`. Isso significa que o valor será tratado como um número sem sinal, mantendo os mesmos bits. Portanto, a representação interna de `ui` será a mesma que a de `sc`.
Programa de Teste:
#include <stdio.h>

```
void dump (void *p, int n) {
  unsigned char *p1 = p;
  while (n--) {
    printf("%p - %02x\n", p1, *p1);
    p1++;
  }
}

int main (void) {
  signed char sc = -1;
  unsigned int ui = sc;
  dump(&ui, sizeof(ui));
  return 0;
}
```

Se você compilar e executar este programa, verá a representação em memória de `ui` em hexadecimal. O resultado será `ffffffff`.