In these questions, your machine is little endian. Here are the prototypes of C procedures:

```
char *strcpy(char *dest, const char *src);
char *strcat(char *dest, const char *src);
char *memcpy(void *dest, const void *src, int bytes);
size_t strlen(char *s);
```

Question 1: If you are lucky enough to have a segmentation violation on this code, which line will cause the segmentation violation?

```
/* Line  1 */  int main()
/* Line  2 */  {
/* Line  3 */    char s[100];
/* Line  4 */    int *vec;
/* Line  5 */    void *v[100];
/* Line  6 */    void *x[100];
/* Line  7 */    int i;
/* Line  8 */
/* Line  9 */    strcpy(s, "");
/* Line 10 */    for (i = 0; i < 50; i++) {
/* Line 11 */      strcat(s, "j");
/* Line 12 */      vec[i] = i;
/* Line 13 */      v[i] = (void *) i;
/* Line 14 */      x[i] = (void *) &i;
/* Line 15 */    }
/* Line 16 */    return 0;
/* Line 17 */  }
```

Question 2: What is the output of this program?

```
int main()
{
  char b[30];
  int i;

  strcpy(b, "ABCDEFGHIJKLMNOPRSTUVWXYZ");
  i = ('C' << 16) | ('D' << 8) | 'E';
  memcpy(b+2, &i, 4);
  printf("%d\n", (int) strlen(b));
  return 0;
}
```

Question 3: What is the 1st line of output of the following program?
Question 4: What is the 2nd line of output of the following program?

```
int main()
{
  unsigned char p[8];
  unsigned int *x;

  x = (unsigned int *) p;
  p[0] = (0x7 << 4) | 0xc;
  p[1] = (0xa << 4) | 0x9;
  p[2] = (0x3 << 4) | 0x2;
  p[3] = (0xf << 4) | 0xe;
  x[1] = 0x15768;

  printf("0x%x\n", x[0]);
  printf("%x-%x-%x-%x\n", p[4], p[5], p[6], p[7]);
  return 0;
}
```

Answers to the clicker questions

Question 1: Line 12 -- **vec** is an uninitialized variable. This is one of the reasons I urge you to set your pointers to NULL at the beginning of your procedures. That will help you find bugs like this, because you're guaranteed to generate a seg fault.

Question 2: Since your machine is little endian, the four bytes of **i** are going to be 'E', 'D', 'C' and 0, in that order. It's a three-character string "EDC". That means that after the **memcpy()**, **b** becomes "ABEDC*GHI....", where the asterisk is the null character. The answer is five.

Question 3. Let's suppose that the value of **p** is 0x12000. Here's memory:

Address	+3	+2	+1	+0
0x1200	0xfe	0x32	0xa9	0x7c
0x1204	0x00	0x01	0x57	0x68

The answer is 0xfe32a97c.

Question 4: From the picture above: 68-57-1-0