Pointers and longs are 4 bytes. The machine is little endian. So, if an integer is 0xabcdef88, then its first byte is 0x88, and its last byte is 0xab. There are no segmentation violations or bus errors in any of this code.

Here are prototypes of `strcpy()`, `strcat()` and `memcopy()`:

```c
char *strcpy(char *to, char *from);
char *strcat(char *to, char *from);
void *memcpy(void *to, void *from, int number_of_bytes);
```

---

**Question 1**

What is the output of the program below:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    unsigned char c, d;
    unsigned int i, j, k, n, m;
    i = 0x5746812;
    j = 0x78210735;
    k = 0x0f0f0f0f;
    n = 0x57440000;
    m = 0x6812;
    c = (i & 0xff);
    d = (c << 4);
    printf("1: 0x%08x\n", (n << 3));
    printf("2: 0x%08x\n", (m << 2));
    printf("3: 0x%08x\n", i & k);
    printf("4: 0x%08x\n", i & (k << 4));
    printf("5: 0x%08x\n", j | k);
    printf("6: 0x%08x\n", i >> 16);
    printf("7: 0x%08x\n", j << 12);
    printf("8: 0x%08x\n", c);
    printf("9: 0x%08x\n", d);
    printf("A: 0x%08x\n", i ^ n);
    printf("B: 0x%08x\n", n + 16);
}
```

---

**Question 2**

What is the output of the program below:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char buf[50];
    char *x, *y, *z;
    x = buf;
y = buf+10;
z = buf+20;
strcpy(y, "ABCDEF");
strncpy(z, "ABCDEF", 3);
strcat(y, ":");
*y = '\0';
z[5] = '\0';
for (x = buf; x - buf < 35; x += 7) {
    printf("%s\n", x);
}
}
```

---

**Work Space**

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
Pointers and longs are 4 bytes. The machine is little endian. So, if an integer is 0xabcdef88, then its first byte is 0x88, and its last byte is 0xab. There are no segmentation violations or bus errors in any of this code.

Question 3: What is the output of this program?

```c
int pm(int *p)
{
    char *s;
    char **t;

    s = (char *) p[3];
    printf("1. %s\n", s);

    s = (char *) p;
    s[0]++;
    s[3] = 'T';
    printf("2. %s\n", s);
    printf("3. 0x%08x\n", *p);
    printf("4. 0x%08x\n", p[1] | p[2]);
    printf("5. 0x%08x\n", p[3] ^ p[4]);

    t = (char **) &p[5];
    printf("6. 0x%08x\n", (unsigned int) (*t));
    printf("7. %s\n", *t);
}
```

p is 0x572b20

<table>
<thead>
<tr>
<th>Address:</th>
<th>Hex</th>
<th>Chars</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x572b20:</td>
<td>0x00572b54</td>
<td>'T'</td>
</tr>
<tr>
<td>0x572b24:</td>
<td>0x00572b30</td>
<td>'0'</td>
</tr>
<tr>
<td>0x572b28:</td>
<td>0x00572b4c</td>
<td>'L'</td>
</tr>
<tr>
<td>0x572b2c:</td>
<td>0x00572b38</td>
<td>'B'</td>
</tr>
<tr>
<td>0x572b30:</td>
<td>0x00572b44</td>
<td>'D'</td>
</tr>
<tr>
<td>0x572b34:</td>
<td>0x00572b3c</td>
<td>'&lt;'</td>
</tr>
<tr>
<td>0x572b38:</td>
<td>0x00572b40</td>
<td>'@'</td>
</tr>
<tr>
<td>0x572b3c:</td>
<td>0x00572b24</td>
<td>'S'</td>
</tr>
<tr>
<td>0x572b40:</td>
<td>0x00572b28</td>
<td>'('</td>
</tr>
<tr>
<td>0x572b44:</td>
<td>0x00572b48</td>
<td>'H'</td>
</tr>
<tr>
<td>0x572b48:</td>
<td>0x00572b34</td>
<td>'4'</td>
</tr>
<tr>
<td>0x572b4c:</td>
<td>0x00572b50</td>
<td>'P'</td>
</tr>
<tr>
<td>0x572b50:</td>
<td>0x00572b20</td>
<td>'+'</td>
</tr>
<tr>
<td>0x572b54:</td>
<td>0x00572b58</td>
<td>'X'</td>
</tr>
<tr>
<td>0x572b58:</td>
<td>0x00572b2c</td>
<td>','</td>
</tr>
</tbody>
</table>
Pointers and longs are 4 bytes. The machine is little endian. So, if an integer is 0xabcdef88, then its first byte is 0x88, and its last byte is 0xab. There are no segmentation violations or bus errors in any of this code.

**Question 4:**

When the following procedure is called, the value of $x$ is 0x68423fe0, and the contents of the 64 bytes starting with $x$ are as follows. I'm printing sets of four bytes in hexadecimal and in decimal.

<table>
<thead>
<tr>
<th>Address</th>
<th>Hexadecimal</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x68423fe0</td>
<td>0x68423ffe0</td>
<td>1749172220</td>
</tr>
<tr>
<td>0x68423fe4</td>
<td>0x68423fe0</td>
<td>1749172192</td>
</tr>
<tr>
<td>0x68423fe8</td>
<td>0x68423ffe0</td>
<td>1749172220</td>
</tr>
<tr>
<td>0x68423ffec</td>
<td>0x68423ff4</td>
<td>1749172212</td>
</tr>
<tr>
<td>0x68423ff0</td>
<td>0x68424014</td>
<td>1749172244</td>
</tr>
<tr>
<td>0x68423ff4</td>
<td>0x68423fe0</td>
<td>1749172192</td>
</tr>
<tr>
<td>0x68423ff8</td>
<td>0x68423fe0</td>
<td>1749172200</td>
</tr>
<tr>
<td>0x68423ff4</td>
<td>0x68424008</td>
<td>1749172232</td>
</tr>
<tr>
<td>0x68424000</td>
<td>0x68423fe8</td>
<td>1749172200</td>
</tr>
<tr>
<td>0x68424004</td>
<td>0x68424004</td>
<td>1749172228</td>
</tr>
<tr>
<td>0x68424008</td>
<td>0x68424014</td>
<td>1749172244</td>
</tr>
<tr>
<td>0x6842400c</td>
<td>0x68423ffe0</td>
<td>1749172220</td>
</tr>
<tr>
<td>0x68424010</td>
<td>0x68423fe0</td>
<td>1749172200</td>
</tr>
<tr>
<td>0x68424014</td>
<td>0x68424004</td>
<td>1749172228</td>
</tr>
<tr>
<td>0x68424018</td>
<td>0x68424000</td>
<td>1749172224</td>
</tr>
</tbody>
</table>

Your job is to tell me the output of the following procedure:

```c
#include <sys/mman.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void a(unsigned int *x)
{
    unsigned int **e, ***f;
    e = (unsigned int **) x;
    f = (unsigned int ***) x;

    printf("0: %d
", *x);
    printf("1: %d
", *x);
    printf("2: %d
", **(e[0]));
    printf("3: %d
", ***f);
    printf("4: %d
", (*(int *)(f[0])));
    printf("5: %d
", ***f);
    printf("6: 0x%x
", (int *)(e[0]));
    printf("7: 0x%x
", f[0][1][1]);
    printf("8: 0x%x
", f[1][1][1]);
    printf("9: 0x%x
", *(f[12]) - *(f[10]));
}```