For the next three questions, consider the following short program:

```c
#include <stdio.h>

int main() {
    char* pC = NULL;               // Line 1
    pC = malloc( sizeof( char ) );  //      2
    *pC = 'x';                     //      3
    return 0;
}
```

1. Which of the following best describes the situation after Line 1 has been executed?
   1) pC does not have a target, and pC does not have a known value.
   2) pC does not have a target, but pC does have a known value.
   3) pC has a target, but its target does not have a known value.
   4) pC has a target, and its target does have a known value.
   5) None of these

2. Which of the following best describes the situation after Line 2 has been executed?
   1) pC does not have a target, and pC does not have a known value.
   2) pC does not have a target, but pC does have a known value.
   3) pC has a target, but its target does not have a known value.
   4) pC has a target, and its target does have a known value.
   5) None of these

3. Which of the following best describes the situation after Line 3 has been executed?
   1) pC does not have a target, and pC does not have a known value.
   2) pC does not have a target, but pC does have a known value.
   3) pC has a target, but its target does not have a known value.
   4) pC has a target, and its target does have a known value.
   5) None of these

4. Assume the variable declarations:

   ```c
   int Foo = 0;
   int *ptr = &Foo;
   ```

   Which of the following statements will change the value of Foo to 1?
   1) ptr++;                   5) All of these
   2) Foo++;                   6) 1 and 2 only
   3) (*Foo)++;                7) 1 and 4 only
   4) (*ptr)++;                8) 2 and 4 only
   9) 3 and 4 only
   10) None of these
For the next three questions, assume that P and Q are pointers of the same type, and that each has been assigned a value.

5. What comparison would be true if and only if P and Q have targets with the same value?

1) &P == &Q  
2) *P == *Q  
3) P == Q

4) All of them  
5) 1 and 2 only  
6) 1 and 3 only

7) 2 and 3 only
8) None of these

6. What comparison would be true if and only if P and Q have the same target?

1) &P == &Q  
2) *P == *Q  
3) P == Q

4) All of them  
5) 1 and 2 only  
6) 1 and 3 only

7) 2 and 3 only
8) None of these

7. What comparison would be true if and only if P and Q store the same value?

1) &P == &Q  
2) *P == *Q  
3) P == Q

4) All of them  
5) 1 and 2 only  
6) 1 and 3 only

7) 2 and 3 only
8) None of these

For the next three questions, assume that we have a dynamically allocated array A of integers of dimension Size, with memory layout as shown:

```c
const int Size = 5;
int *A = malloc(Size * sizeof(int));
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>006601B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006601B4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006601BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006601BC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006601C0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. Which code fragment(s) could be inserted in the blank in order to safely initialize each element of A to zero?

```c
int* p = &A[0];
for (int Idx = 0; Idx < Size; Idx++, p++) {
  ______________;
}
```

1) *A = 0;  
2) A[Idx] = 0;  
3) *p = 0;

4) *Idx = 0;  
5) All of the above  
6) 1 and 2 only

7) 1 and 3 only  
8) 2 and 3 only  
9) 1 and 4 only  
10) None of these

9. What value will be be assigned to A[3] by the code fragment:

```c
for (int Idx = 0; Idx < Size; Idx++) {
  A[Idx] = (int) &A[Idx];  // typecast converts address to int
}
```

1) 006601B0  
2) 006601B4  
3) 006601BB

4) 006601BC  
5) 006601C0  
6) Unknown

7) None of these
10. What logical error(s) would result if the following statement were executed:

```c
const int Size = 5;
int *A = malloc(Size * sizeof(int));
A = malloc(2 * Size * sizeof(int));
```

1) A dangling pointer would result (a pointer whose value is the address of memory that the program no longer owns).
2) A memory leak would result (the program would own memory that it could no longer access).
3) Both a dangling pointer and a memory leak would result.
4) Neither a dangling pointer nor a memory leak, but some other logical error would result.
5) No logical error would result.