I compile the following program to **a.out**:

```
#include <vector>
#include <iostream>
#include <cstdio>
using namespace std;

int main(int argc, char **argv)
{
  vector <string> args;
  int x;
  int i;

  for (i = 1; i < argc; i++) args.push_back(argv[i]);
  cin >> hex >> x;   // This reads an integer in hex to the variable x

  for (i = 0; i < args.size(); i++) {
     if (x & (1 << i)) cout << args[i];
  }
  cout << endl;
  return 0;
}
```

Question 1: What is the output of the following:

```
UNIX> echo 0x3 | ./a.out a b c d
```

Question 2: What is the output of the following:

```
UNIX> echo 0xc | ./a.out a b c d
```

Question 3: What is the output of the following:

```
UNIX> echo 0xa5 | ./a.out a b c d e f g h
```

Clicker Question Answers

This program puts its command line arguments into a vector. It then reads an integer in hex format from standard input. It then runs through the indices of the vector, and checks to see if each index's bit is set in the integer. If so, it prints the argument from the vector. At the end, it prints a newline.

Question 1: It reads 3 into **x**. Only bits 0 and 1 of three are set, so it prints the first two command line arguments:

```
UNIX> echo 0x3 | ./a.out a b c d
ab
UNIX>
```

Question 2: Now it reads 0xc into **x**. Bits 2 and 3 of 0xc are set, so it prints the last two command line arguments:

```
UNIX> echo 0xc | ./a.out a b c d
cd
UNIX>
```

Question 3: Now, the vector is composed of 8 strings. We read 0xa5 into **x**. The bits set in **x** are bits 0, 2, 5 and 7 (In binary, **x** is 1010 0101). So:

```
UNIX> echo 0xa5 | ./a.out a b c d e f g h
acfh
UNIX>
```