1) Binary Search: This C program compiles and runs. What is its output?

```c
#include <stdio.h>

int binarySearch(int x, int v[], int high)
{
    int mid;
    int low = 0;

    while (low <= high)
    {
        mid = (low+high)/2;
        printf("[%d %d] ", low, high);

        if (x < v[mid]) high = mid-1;
        else if (x > v[mid]) low = mid+1;
        else return mid;
    }
    return -1;
}

void main(void)
{
    int nums[]={12, 33, 45, 47, 53, 55, 59, 73, 91, 93};
    int n = sizeof(nums)/sizeof(int);
    printf("idx = %d\n", binarySearch(47, nums, n));
    printf("idx = %d\n", binarySearch(88, nums, n));
}```
2) Bit Operators: This C program compiles and runs. What is its output?

```c
#include <stdio.h>

void main(void)
{
    unsigned char x = 73;

    unsigned char a = x << 4;
    unsigned char b = x >> 4;
    unsigned char c = x & 7;
    unsigned char d = x & 99;
    unsigned char e = x | 7;
    unsigned char f = x ^ 7;

    printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n", a, b, c, d, e, f);
}
```
3) **Find Substring - using Pointers.** This C program compiles and runs. What is its output?

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

char *findSubstring(char *str, char *needle) {
    int len = strlen(needle);
    int n = 0;
    while (*str) {
        printf("%c%c\n",*str, *needle);
        if ( *(needle+n) == *str) {
            n++;
            if (n == len) return (str-len)+1;
        } else {
            str -= n;
            n = 0;
        }
        str++;
    }
    return NULL;
}

void main(void) {
    char* sub=findSubstring("Axffoffoofoo","foo");
    printf("==>\%s\n",sub);
}
```
4) This C program compiles and runs. If the output from lines 7 and 8 is:

```
sizeof(long)=8  x=0x7fff29af6530, x[0]=22
```

Then what is the output from line 10?

```c
#include <stdio.h>

void main(void)
{
    long a[] = {22, 33, 44};
    long *x = a;
    printf("sizeof(long)=%lu ", sizeof(long));
    printf("x=%p, x[0]=%d\n", x, x[0]);
    x = x + 2;
    printf("x=%p, x[0]=%d\n", x, x[0]);
}
```

5) This C program compiles and runs. If executed with the command:
```
a.out 00110023
```

Then what is the output?

```c
#include <stdio.h>

void main(int argc, char *argv[])
{
    if (argc == 2)
    {
        int n = 0;
        char *c_pt = argv[1];
        while (*c_pt)
        {
            if (*c_pt < '0' || *c_pt > '1') break;
            n = n*2 + *c_pt-'0';
            c_pt++;
        }
        printf("%d\n", n);
    }
}
```
6) This C program compiles and runs. What is the output?

```c
1) #include <stdio.h>
2) struct Point {int x; int y;};
3) void addToPoint(struct Point *a, struct Point b, int n)
4) {
5)    (*a).x += n;
6)    a->y   += n;
7)    b.x    += n;
8)    b.y    += n;
9) }
10) 
11) void main(void)
12) {
13)    struct Point p1 = {7, 7};
14)    struct Point p2 = {11, 11};
15)    addToPoint(&p1, p2, 2);
16)    printf("p1=(%d, %d)\n", p1.x, p1.y);
17)    printf("p2=(%d, %d)\n", p2.x, p2.y);
18) }
```

7) Memory Allocation - malloc. What expression should be placed inside the call to malloc to allocate memory for an `double` array with `n` rows of `m` columns?

```
double *x = malloc( ? );
```