1. List five keywords that are used to control the “flow” of the program in C. (5)

2. What is the operator <<= used for? Write an expression that uses it and show another way to write an expression with the same meaning. (5)

3. A variable of type void * can be very useful. Why? (5)

4. C programming is said to be close to the machine. One remnant of assembly programming is that C contains a goto keyword that actually works. Why is the goto instruction considered by many hazardous to use? (5)
5. What is the output of this program?

```c
#include <stdio.h>

int x = 8;

int foo(int n)
{
    int y = 4;
    x /= 2;
    y /= 2;
    n -= x + y;
    printf("foo: x=%d, y=%d, n=%d\n", x, y, n);
    return n;
}

void main(void)
{
    int x, n;
    n = 10;
    x = foo(n);
    printf("main: n=%d, x=%d\n", n, x);
    x = foo(n);
    printf("main: n=%d, x=%d\n", n, x);
}
```
6. What is the output of this program?

```c
#include <stdio.h>

void main(void)
{
    unsigned char x = 43;
    unsigned char a = x << 2;
    unsigned char b = x >> 2;
    unsigned char c = x & 31;
    unsigned char d = x & 121;
    unsigned char e = x | 31;
    unsigned char f = x ^ 31;

    printf("a=%d\n", a);
    printf("b=%d\n", b);
    printf("c=%d\n", c);
    printf("d=%d\n", d);
    printf("e=%d\n", e);
    printf("f=%d\n", f);
}
```
7. What is the output of this program?

```c
#include <stdio.h>

void main(void)
{
    char data[] = "boastBOAST";
    data[1] = 'e';
    char *linePt = &data[6];
    *linePt = 'L';
    printf("[%s], [%s]\n", data, linePt);
}
```
8. What is the output of this program?

```c
#include <stdio.h>

struct Point
{
    int x;
    int y;
};

struct Point foo(struct Point p1, struct Point *p2)
{
    p1.x += p2->x;
    p2->y += p1.y;
    return p1;
}

void main(void)
{
    struct Point a = {1, 3};
    struct Point b = {5, 4};
    struct Point c = foo(a, &b);

    printf("a=(%d, %d)\n", a.x, a.y);
    printf("b=(%d, %d)\n", b.x, b.y);
    printf("c=(%d, %d)\n", c.x, c.y);
}
```
9. What is the output of this program?

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

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

void main(void)
{
    findSubstring("ACBCDCE", "CD");
}
```
10. What is the output of this program?

```c
#include <stdio.h>

int binarySearch(int x, int v[], int length)
{
    int low, high, mid;
    low = 0;
    high = length -1;

    while (low <= high)
    {
        mid = (low+high)/2;
        printf("[%d %d %d] ", low, mid, 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, 13, 15, 17, 21, 23, 27, 39, 43, 51};
    printf("index = %d\n", binarySearch(43, nums, 10));
    printf("index = %d\n", binarySearch(10, nums, 10));
}
```
11. What is the output of this program?

```c
#include <stdio.h>
void swap(int v[], int i, int j)
{
    int c = v[i];
    v[i] = v[j];
    v[j] = c;
}

void quicksort(int v[], int left, int right)
{
    int i, last;
    printf("[%d, %d]\n", left, right);
    if (left >= right) return;

    swap(v, left, (left+right)/2);
    last = left;
    for (i=left+1; i <= right; i++)
    {
        if (v[i] < v[left])
        {
            last++;
            swap(v, last, i);
        }
    }

    swap(v, left, last);
    quicksort(v, left, last-1);
    quicksort(v, last+1, right);
}

void main(void)
{
    int v[] = {5, 2, 7, 8, 3, 1};

    int arraySize = sizeof(v)/sizeof(int);
    quicksort(v, 0, arraySize-1);
}
```