Sample Question 1.

Consider this C program:

```c
#include <stdio.h>

int main( void )
{
    int x=1;
    int y=0;

    while( x < 100 ) {
        x = x + 8 * y++ ;
        printf("%d\n", x);
    }

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.
Sample Question 2.

Consider this C program:

```
#include <stdio.h>

int main( void )
{
    int *p, *q;
    int x, y;

    x = 7;
    y = 8;

    q = &x;
    *q = 10;
    p = q;

    y = *p + *q;
    *p = x + y;

    printf("x = %d, y = %d\n", x, y);

    return 0;
}
```

The program is valid C. It executes without error. Indicate clearly and exactly what output will be printed.
Sample Question 3.

Consider this C program:

```c
#include <stdio.h>

int df( int n )
{
    if( n < 2 ) {
        return( 1 );
    }
    else {
        return( n * df( n-2 ));
    }
}

int main( void )
{
    printf("5!! = %d\n", df( 5 ));
    printf("6!! = %d\n", df( 6 ));

    return 0;
}

The program is valid C. It executes without error. Indicate clearly and exactly what output will be printed.
Sample Question 4.

Consider this C program:

#include <stdio.h>

void print_edge( int k )
{
    int i;

    for( i=0; i < k; i++ ) {
        printf( "+-" );
    }
    printf( "+
" );
}

void print_legs( int k )
{
    int i;

    for( i=0; i < k; i++ ) {
        printf( "| " );
    }
    printf( "|
" );
}

int main( void )
{
    int k;

    for( k=0; k < 4; k++ ) {
        print_edge( k );
        print_legs( k );
    }
    print_edge( k );

    return 0;
}

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.
Sample Question 5.

Consider this C program:

```c
#include <stdio.h>

void f( char *s )
{
    if( ! *s ) {
        return;
    }
    f( s+1 );
    putchar( *s );
}

int main(void)
{
    f("kernighan");
    putchar(‘
’);
    return 0;
}
```

The program is valid C. It executes without error. Indicate clearly and exactly what output will be printed.
Sample Question 6.

Consider this C program:

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

struct node {
    int value;
    struct node *next;
};

struct node *new( int val )
{
    struct node *n = malloc(sizeof(struct node));
    n->value = val;
    n->next = NULL;
    return n;
}

struct node *insert( struct node *n, struct node *h )
{
    n->next = h;
    return n;
}

int main( void )
{
    struct node *a, *b, *c;
    int i;

    a = new(1);
    b = insert( new(5), insert( new(2), a ));
    c = insert( new(3), b->next );
    a = insert( a, c );

    for( i=0; i < 5; i++ ) {
        printf(" %d", b->value );
        b = b->next;
    }

    printf("\n");

    return 0;
}
```

The program is valid C. It executes without error. Indicate clearly and exactly what output will be printed.
Sample Question 7.

Consider this C program:

```c
#include <stdio.h>

int a = 1;
static int b = 1;

int f( int c )
{
    static int d = 1;
    int e = 0;

    a++;
    b += d;
    c = c + 2;
    d = d + a - b + c;
    e = e + 2*d + 1;
    return( e+2 );
}

int main( void )
{
    int a, d;
    a = 3;

    for( d=0; d < 3; d++ ) {
        printf("%d\n", f(a));
    }
    printf("%d\n", a );
    printf("%d\n", b );
    printf("%d\n", d );

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.
Sample Question 8.

a. Convert the following binary number to decimal.

\[10101.001_2\]

b. Convert the following decimal number to binary.

\[45.375_{10}\]

c. Assume that signed binary numbers are stored in two’s complement form, in 8 bits (for example, \(11111111_2 = -1_{10}\)).

(i) Convert the following binary number to decimal.

\[01101011_2\]

(ii) What is the negative of the number from part c(i), written in two’s complement (binary) form?

(iii) Use your answer from part c(ii) to compute the result of the following binary subtraction. Write your answer in two’s complement form.

\[00110001_2 - 01101011_2\]

The instruction set for the simple machine presented in lectures is given in the table on the next page. The following three sub-questions refer to the program shown above the table (also on the next page). All addresses and memory contents are given in hexadecimal (base 16) notation.

d. What would be in registers R1, R2, R3 and R4 after running the program starting at address A0, given that the relevant memory contents are as shown?

e. Suppose that the contents of memory location 81 is changed from 02 to 03. What would now be in register R3 after running the above program starting at address A0?

f. Write a segment of C code that accomplishes the same task as this machine language program.
Machine Language Program

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>00</td>
</tr>
<tr>
<td>81</td>
<td>02</td>
</tr>
<tr>
<td>A0</td>
<td>20 00</td>
</tr>
<tr>
<td>A2</td>
<td>21 01</td>
</tr>
<tr>
<td>A4</td>
<td>12 81</td>
</tr>
<tr>
<td>A6</td>
<td>23 01</td>
</tr>
<tr>
<td>A8</td>
<td>B2 B4</td>
</tr>
<tr>
<td>AA</td>
<td>50 01</td>
</tr>
<tr>
<td>AC</td>
<td>40 34</td>
</tr>
<tr>
<td>AE</td>
<td>53 34</td>
</tr>
<tr>
<td>B0</td>
<td>53 34</td>
</tr>
<tr>
<td>B2</td>
<td>B0 A8</td>
</tr>
<tr>
<td>B4</td>
<td>33 80</td>
</tr>
<tr>
<td>B6</td>
<td>C0 00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Operand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RXY</td>
<td>LOAD register R with bit pattern found in memory cell whose address is XY</td>
</tr>
<tr>
<td>2</td>
<td>RXY</td>
<td>LOAD register R with the bit pattern XY</td>
</tr>
<tr>
<td>3</td>
<td>RXY</td>
<td>STORE the bit pattern found in register R in the memory cell whose address is XY</td>
</tr>
<tr>
<td>4</td>
<td>0RS</td>
<td>MOVE the bit pattern found in register R to register S</td>
</tr>
<tr>
<td>5</td>
<td>RST</td>
<td>ADD the bit patterns in registers S and T as though they were two's complement representations, and leave the result in register R</td>
</tr>
<tr>
<td>6</td>
<td>RST</td>
<td>ADD the bit patterns in registers S and T as though they represented values in floating point notation, and leave the floating point result in register R</td>
</tr>
<tr>
<td>7</td>
<td>RST</td>
<td>OR the bit patterns in registers S and T and place the result in register R</td>
</tr>
<tr>
<td>8</td>
<td>RST</td>
<td>AND the bit patterns in registers S and T and place the result in register R</td>
</tr>
<tr>
<td>9</td>
<td>RST</td>
<td>EXCLUSIVE-OR the bit patterns in registers S and T and place the result in register R</td>
</tr>
<tr>
<td>A</td>
<td>R0X</td>
<td>ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low order end to the high order end.</td>
</tr>
<tr>
<td>B</td>
<td>RXY</td>
<td>JUMP to the instruction located in the memory cell at the address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution.</td>
</tr>
<tr>
<td>C</td>
<td>000</td>
<td>HALT execution</td>
</tr>
</tbody>
</table>