Question 1

```c
#include <stdio.h>

// Question 1

int main(void) {
    int a[]={1,2,3,4,5}, i, cnt=0;

    for (i=0;i<5;i++){
        a[i] += cnt;
        cnt++;
    }

    for (i=0;i<5;i++)
        printf("%i\n", a[i]);

    system("PAUSE");
    return 0;
}
```

Three integer variables: an array **a** with 5 five values, **i** and **cnt**.

1st Loop

The first loop iterates through the values of the array **a**. During each iteration, the current value of **cnt** is added to the item of the array with index **i**.

1st Iteration

Variables: **i** = 0, **cnt** = 0, a[**i**] = 1 (the first value in the array), a[1,2,3,4,5]

a[**i**] += **cnt**; (**cnt** is zero, therefore value in the array is not changed)

cnt++; (the value of count is incremented, it is now 1)

End of iteration: a[1,2,3,4,5]

2nd Iteration

Variables: **i** = 1, **cnt** = 1, a[**i**] = 2 (the second value in the array), a[1,2,3,4,5]

a[**i**] += **cnt**; (**cnt** is 1, therefore 1 is added to the value in the array, it is now 3)

cnt++; (the value of count is incremented, it is now 2)

End of iteration: a[1,3,3,4,5]

3nd Iteration

Variables: **i** = 2, **cnt** = 2, a[**i**] = 3 (the third value in the array), a[1,3,3,4,5]

a[**i**] += **cnt**; (**cnt** is 2, therefore 2 is added to the value in the array, it is now 5)

cnt++; (the value of count is incremented, it is now 3)

End of iteration: a[1,3,5,4,5]

4th Iteration

End of iteration: a[1,3,5,7,5]

5th Iteration

End of iteration: a[1,3,5,7,9]

2nd Loop

This loop prints all the values in the array **a**. Each one is placed on a new line (\n).

Question 2

```c
#include <stdio.h>

// Question 2

int main(void) {
    int x=4562;
    int a[4], i;
    int idx=0;
    int g=0;

    while(x>0){
        a[idx] = x%10;
        x /= 10;
        idx++;
    }

    for(i=0;i<4;i++){
        g *= 10;
        g += a[i];
    }

    printf("Value is: %i\n",g);
    system("PAUSE");
    return 0;
}
```

All variables are integers. Importantly note that **a** is an empty array with places to hold four integer values.

1st Loop

The first loop continues while the value of **x** is greater than zero. In the first line, the value at index **idx** in the array is set to the remainder of **x** divided by 10. Then, in the second line, the value of **x** is set equal to **x** divided by 10. In the third line the value of **idx** is incremented by 1.

1st Iteration

Variables: **x** = 4562, **idx** = 0, a = []

a[**idx**] = **x**%10; (x divided by 10 is 456.2. idx is set to the remainder, which is 2)

x /= 10 (x is set to itself divided by 10: 4562/10 = 456)

idx++; (the value of idx is incremented, it is now 1)

End of iteration: a[2]

2ndIteration

Variables: **x** = 456, **idx** = 1, a = [2]

a[**idx**] = **x**%10; (x divided by 10 is 45.6. idx is set to the remainder, which is 6)

x /= 10 (x is set to itself divided by 10: 456/10 = 45)

idx++; (the value of idx is incremented, it is now 2)

End of iteration: a[2,6]

3rd Iteration

End of iteration: a[2,6,5]

4th Iteration

End of iteration: a[2,6,5,4]

2nd Loop

The second loop iterates four times. Firstly it multiplies **g** by ten. Then it adds the value at index **i** in the array **a** to **g**.

1st Iteration

Variables: **i** = 0, **g** = 0, a = [2,6,5,4]

g *= 10; (g is set to itself multiplied by 10: 0 * 10 = 0)

g += a**[i]**; (the value at index i in the array is added to g: 0 + 2 = 2)

End of iteration: g = 2

2nd Iteration

End of iteration: g = 26

3rd Iteration

End of iteration: g = 265

4th Iteration

End of iteration: g = 2654

When the second loop is complete, the program prints the value of **g**.

Question 3

```
1  #include <stdio.h>
2
3  // Question 3
4
5  int main(void) {
6      int a=0xfa, b=0xe4,c;
7
8      c=f1(&a,&b);
9      printf("Value is %x\n",c);
10
11     system("PAUSE");
12     return 0;
13 }
14
15 f1(int *p1, int *p2)
16 {
17     int x = ((*p1&*p2)|1)^*p1;
18
19     return x;
20 }
```

All variables are integers. **a** and **b** are declared in hexadecimal format. There is one function named **f1** which is declared on Line 15 above.

At Line 8 above, the main program indicates that it wants to utilise **f1**. This function requires two integer values which must be passed as pointers. The use of **&a** and **&b** tell the function to work with the data stored in **a** and **b**, and to access it using pointers.

StackOverflow has a more detailed explanation of pointers at this link: http://stackoverflow.com/questions/2094666/pointers-in-c-when-to-use-the-ampersand-and-the-asterisk

Function f1

The function calculates a value for **x.** The statement looks complicated but becomes much simpler once broken into smaller parts (by using precedence rules). Work on what's inside the brackets first.

The statement uses binary operations – convert **a** and **b** to binary. The use of 0x indicates that they are hexadecimal:

a = 0xfa, the binary value of fa is 11111010

b = 0xe4, the binary value of e4 is 11100100

The statement: ((*p1&*p2)|1)^*p1

Part 1: **p1&*p2 [in English: *p1 AND *p2]*

*p1	11111010
*p2	11100100
*p1&*p2	11100000

Part 2: *((*p1&*p2)|1) [in English: ((*p1 AND *p2) OR 1)]*

*p1&*p2	11100000
1	00000001
*((*p1&*p2)|1)*	11100001

Part 3: *((*p1&*p2)|1)^*p1 [in English: ((*p1 AND *p2) OR 1) XOR *p1]*

*((*p1&*p2)|1)*	11100001
*p1	11111010
*((*p1&*p2)|1)^*p1*	00011011

When the function is complete the final result is returned, and then stored in the variable **c**. The value of **c** is printed. The usage of %x indicates that the value should be printed in hexadecimal format.

Therefore, the binary value 00011011 is output as 1b.

Question 4

```
1  #include <stdio.h>
2  #include <string.h>
3
4  // Question 4
5
6  int main(void) {
7      char a[]={"abcdefg"};
8
9      f1(a);
10     f2(a);
11
12     system("PAUSE");
13     return 0;
14 }
15
16 f2(char *p)
17 {
18     int len=strlen(p),i;
19
20     for(i=0;i<len-1;i++)
21         p++;
22
23     for(i=0;i<len;i++){
24         printf("%c\n",*p);
25         p--;
26     }
27 }
28
29 f1(char *p)
30 {
31     int len=strlen(p),i;
32
33     for(i=0;i<len;i++){
34         *p += 1;
35         p++;
36     }
37 }
```

This program uses two functions to perform operations on the contents of the array **a**. The array initially contains 7 values – a, b, c, d, e, f, g.

The main program passes control to the function f1 on Line 9 above. The function begins on Line 29. It takes one argument – a pointer to a char value. In this case, the pointer is to the array **a**.

Function f1

Variables: **len** = 7 (the length of the array), **i**, **a**[a,b,c,d,e,f,g]

The loop iterates 7 times (controlled by **len**).

**p += 1* (in each iteration, the char value at memory location p is incremented by 1. For example, during the first iteration a becomes b, during the second iteration b becomes c, and so on.)

p++ (the pointer is moved to the next memory location)

At the completion of the loop, the arrays values are:

A[b,c,d,e,f,g,h]

Control returns to the main program, and is then passed to the function f2 as seen on Line 10 above. The function begins on Line 16. Again, it takes one argument – a pointer to a char value. Again, the pointer is to the array **a**.

Function f2

Variables: **len** = 7, **i**, **a**[b,c,d,e,f,g,h]

The first loop moves the location of the pointer from the beginning of the char array to the end of the char array.

The second loop prints the characters. One character is printed per iteration and each is placed on a new line. Importantly, the characters are printed in reverse order. This is because the previous loop placed the pointer at the end of the array. *p*-- moves the pointer back one position.

Output from the second loop is:

h

g

f

e

d

c

b

Question 5

```
1  #include <stdio.h>
2
3  // Question 5
4
5  int main(void) {
6      int g[]={5,6,7,8,9};
7
8      fn1(g);
9      fn2(g);
10
11     system("PAUSE");
12     return 0;
13 }
14
15 fn2(int *p)
16 {
17     int i;
18
19     for(i=0;i<5;i++){
20         printf("%i\n",*p);
21         p++;
22     }
23 }
24
25 fn1(int *p)
26 {
27     int i,mask=3;
28
29     for(i=0;i<5;i++){
30         *p = *p & mask;
31         mask = mask << 1;
32         p++;
33     }
34 }
```

This program uses two functions to perform operations on the contents of the array **g**. The array initially contains 5 values – 5, 6, 7, 8, 9.

The main program first passes control to the function fn1 on Line 8 above. This function takes a pointer to the array **g** as an argument. The pointer allows the function to directly modify the contents of the array.

Function fn1

Variables: **i** = 0, **mask** = 3

There is one loop in this function which iterates five times. One iteration will be shown to demonstrate how it works – its function is to modify the contents of the array using a mask and binary operations.

Iteration 1

**p = *p & mask;*

This line performs a bitwise AND between the value at the pointer and the mask. The value at the pointer is 5 – the first value in the array. In binary, 5 becomes 0101. The mask 3 becomes 0011. A bitwise AND produces the result 0001, as shown:

*p (5)	0101
mask (3)	0011
Result of AND	0001

The value in the array is changed to the result of the AND operation – 1.

mask = mask << 1

The value of the mask is also updated. It is bitwise shifted to the left. This means all the bits are moved to the left by one.

mask (3) which is 0011 become 0110. This is the number 6.

p++

Finally, the location of the pointer is moved forward by one to point to the next item in the array.

Array at end of iterations: g[1,6,7,8,9]

Iteration 2

*p (6)	0110
mask (6)	0110
Result of AND	0110

The new mask: 0110 becomes 1100, which is 12.

Array contents: g[1,6,7,8,9]

Iteration 3

*p (7)	0111
mask (12)	1100
Result of AND	0100

The new mask: 1100 becomes 11000, which is 24.

Array contents: g[1,6,4,8,9]

Iteration 4

Array contents: g[1,6,4,8,9]

Iteration 5

Array contents: g[1,6,4,8,0]

Following the loop, control returns to the main program, it then calls the function fn2 as seen on Line 9 above. This function loops through all values in the array and prints them out one by one, placing each one on a new line.

Question 6

```c
#include <stdio.h>

// Question 6

int main(void) {
    int i=0xface, j=4;

    fn(i,j);

    system("PAUSE");
    return 0;
}

fn(int a,int b)
{
    while(a>0){
        printf("%x\n",a);
        a = a >> b;
    }
}
```

This program uses the function fn to print the value of the variable **i** in hexadecimal format. It prints the value in a loop, and with each iteration the value is bit shifted to the right. The main program passes two variables to the function fn – these are **i** and **j** as seen above.

Function fn

The while loop will iterate while the value of the variable **a** is greater than 0. The loop first prints the value of the variable **a** in hexadecimal format (%x). Then, it bit-shifts the value of that variable to the right by **b**. **b** is 4. Thus, **a** is shifted 4 places to the right during each iteration of the loop.

Iteration 1

Print output: face

Shift operation: **a** is currently 0xface. In binary, this is: 1111101011001110. The shift operation moves everything four places to the right, which produces: 0000111110101100. In essence, the four right most bits are removed.

Iteration 2

Print output: fac

Shift operation: 0000111110101100 becomes 0000000011111010

Iteration 3

Print output: fa

Shift operation: 0000000011111010 becomes 0000000000001111

Iteration 4

Print output: fa

Shift operation: 0000000000001111 becomes 0000000000000000

a is no longer greater than zero, so the loop does not continue after this point.

Question 7

```c
#include <stdio.h>

// Question 7

f(char *p, char Start, char End);

int main(int argc, char *argv[]) {
    char a[] = {"Hello12 World34567!!???"};
    printf("val is %i\n",f(a,'A','Z'));
    printf("val is %i\n",f(a,'a','z'));
    printf("val is %i\n",f(a,'0','9'));

    system("PAUSE");
    return 0;
}

f(char *p, char Start, char End)
{
    int i, cnt=0, len=strlen(p);

    for(i=0;i<len;i++){
        if (*p >= Start && *p <= End)
            cnt++;
        p++;
    }

    return cnt;
}
```

This program uses the function f to count the number of uppercase, lowercase and numeric characters contained within the char array **a**. There three print statements in the main program. During each statement, the function f is asked to execute.

The first print statement uses the function to count the uppercase characters, the second to count the lowercase characters and the third to count the numeric characters. Each print statement outputs its value on a new line. The final output from the program is:

val is 2

val is 8

val is 7

Function f

The **cnt** variable is used within the for loop to count the number of characters found. The **len** variables hold the length of the character array – it allows the loop to determine how many times it must iterate.

The loop will iterate until it analyses every character in the array. The if statement checks if the character currently being analysed (*p) is between the two other specified characters. For example, when counting the number of uppercase characters, the statement could be interpreted as:

if (*p >= ‘A’ && *p <= ‘Z’)

    increment **cnt**

When the loop completes its iterations, the value of **cnt** is returned (sent back) to the main program.

Question 8

```c
#include <stdio.h>

// Question 8

int main() {
    int values[] = {1,11,9,2,5};

    printf("%i",f(values,5));

    system("PAUSE");
    return 0;
}

// Below is the function you are asked to write
f(int *p, int x)
{
    int i, max=0;

    for(i=0;i<x;i++){
        if(i==0){
            max = *p;
        }
        else if(*p > max){
            max = *p;
        }

        p++; // move to next position in array
    }

    return max;
}
```

Question 9

```
1  #include <stdio.h>
2
3  // Question 9
4
5  int main() {
6      int i, reverse, tempNum;
7
8      for(i=0;i<=1000;i++){
9          reverse = 0;
10         tempNum = i;
11
12         while(tempNum > 0){
13             reverse = reverse * 10;
14             reverse = reverse + tempNum % 10;
15             tempNum = tempNum/10;
16         }
17
18         if(i == reverse){
19             printf("%i\n",i);
20         }
21     }
22
23     system("PAUSE");
24     return 0;
25 }
```

The for loop runs 1000 times. The while loop reverses the number. If the reverse of the number is the same as the number, it is printed out.

While Loop Example (Reverse 121)

i = 121

tempNum = 121

reverse = 0

Iteration 1

Line 13: reverse = 0

Line 14: reverse = 0 + 1 = 1

Line 14: tempNum = 121/10 = 12

Iteration 2

Line 13: reverse = 10

Line 14: reverse = 10 + 2 = 12

Line 14: tempNum = 12/10 = 1

Iteration 3

Line 13: reverse = 120

Line 14: reverse = 120 + 1 = 121

Line 14: tempNum = 1/10 = 0