

## QE1 (Boolean and bitwise operations)

1) Capture and run the following code

```
#include <stdio.h>
#include <stdint.h>

void PrintToBinary8(uint8_t);

int main() {
    uint32_t i;
    uint8_t x_reg;
    /* Boolean and bit wise operations */
    /* Class example */
    uint8_t a = 5 && (13) || 6;
    uint8_t b = 5 & (~3) | 6;
    printf ("The a value is: %x ", a);
    PrintToBinary8(a);
    printf ("The a value is: %x ", b);
    PrintToBinary8(b);

    /* End of main program */
    return(0);
}

void PrintToBinary8(uint8_t num) {
    for(int i=7;i>=0;i--){
        if (num & (1<<i))
            printf("1");
        else
            printf("0");
    }
    printf("\n");
}
```

≡ </> OneCompiler

42qfna88b

NEW C RUN

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4
5 void PrintToBinary8(uint8_t);
6
7
8 int8_t main() {
9     uint32_t i;
10    uint8_t x_reg;
11    /* Boolean and bit wise operations */
12    /* Class example */
13    uint8_t a = 5 || (13) && 6;
14    uint8_t b = (5 | (~3)) & 6;
15    printf ("The a value is: %x ", a);
16    PrintToBinary8(a);
17    printf ("The a value is: %x ", b);
18    PrintToBinary8(b);
19
20    /* End of main program */
21    return(0);
22 }
23 void PrintToBinary8(uint8_t num) {
24     for(int i=7;i>=0;i--){
25         if (num & (1<<i))
26             printf("1");
27         else
28             printf("0");
29     }
30     printf("\n");
31 }
32
```

STDIN

Input for the program ( Optional )

Output:

The a value is: 1 00000001  
The a value is: 4 00000100

2) Given the following statements:

int a = 8 && (!10) || 14;

int b = 8 & (~10) | 14;

a) Obtain the initial values for a and b; Show your procedure to get to these values.

This can be an image with your handwriting.

b) Capture this code and verify your results

d) Report the final code and an image of your results

Handwritten calculations for the bit-wise operations:

Left side (for 'a'):  
8 && (!10) || 14  
8 && (0) || 14  
0 || 14  
1

Right side (for 'b'):  
8 & (~10) | 14  
8 & (~1010) | 14  
8 & (0101) | 14  
1000 & (0101) | 14  
0000 | 1110  
1110 = E

The screenshot shows the OneCompiler IDE with the following C code:

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 void PrintToBinary8(uint8_t);
5
6
7
8 int8_t main() {
9     uint32_t i;
10    uint8_t x reg;
11    /* Boolean and bit wise operations */
12    /* Class example */
13    //uint8_t a = 5 || (13) && 6;
14    //uint8_t b = (5 | (~3)) & 6;
15    int a = 8 && (!10) || 14;
16    int b = 8 & (~10) | 14;
17
18    printf("The a value is: %x ", a);
19    PrintToBinary8(a);
20    printf("The a value is: %x ", b);
21    PrintToBinary8(b);
22
23    /* End of main program */
24    return(0);
25 }
26 void PrintToBinary8(uint8_t num) {
27     for(int i=7;i>=0;i--){
28         if (num & (1<<i))
29             printf("1");
30         else
31             printf("0");
32     }
33     printf("\n");
34 }
35
```

The output of the program is:

```
Output:
The a value is: 1 00000001
The a value is: e 00001110
```

2) Given the following register x\_reg = 0x92:

x_reg	1	0	0	1	0	0	1	0
Bit position	7	6	5	4	3	2	1	0

Write the code to perform the following operations creating masks by shifting bits

- Assume x\_reg = 92, bits 5 and 3 must be set (1) without modifying the others
- Assume x\_reg = 92, bits 7 and 1 must be clear (0) without modifying the others
- Assume x\_reg = 92, bits 4 and 0 must be toggle without modifying others

3) Report the final code and an image of your results

```
OneCompiler
42qfna88b
NEW C RUN
Main.c
1 #include <stdio.h>
2 #include <stdint.h>
3
4 void PrintToBinary8(uint8_t);
5
6
7
8 int main() {
9     uint32_t i;
10    uint8_t x_reg;
11    uint8_t mask;
12
13    x_reg = 0x92;
14    printf("x_reg value before set operation is: %x ", x_reg);
15    PrintToBinary8(x_reg);
16    /* New x_reg value here */
17    mask = (1<<5 | 1<<3);
18    x_reg = x_reg | mask;
19    printf("x_reg value after set operation is: %x ", x_reg);
20    PrintToBinary8(x_reg);
21    /*Operation - clear bit*/
22    x_reg = 0x92;
23    printf("x_reg value before clear operation is: %x ", x_reg);
24    PrintToBinary8(x_reg);
25    /* New x_reg value here */
26    mask = ~(1<<7 | 1<<1);
27    x_reg = x_reg & mask;
28    printf("x_reg value after clear operation is: %x ", x_reg);
29    PrintToBinary8(x_reg);
30    /*Operation - toggle bit*/
31    x_reg = 0x92;
32    printf("x_reg value before toggle operation is: %x ", x_reg);
33    PrintToBinary8(x_reg);
34    /* New x_reg value here */
35    mask = (1<<4 | 1<<0);
36    x_reg = x_reg ^ mask;
37    printf("x_reg value after toggle operation is: %x ", x_reg);
38    PrintToBinary8(x_reg);
39
40
41 /* End of main program */
42 return(0);
43 }
44 void PrintToBinary8(uint8_t num) {
45     for(int i=7;i>=0;i--){
46         if (num & (1<<i)){
47             printf("1");
48         }
49         else
50             printf("0");
51     }
52     printf("\n");
53 }
```

STDIN

Input for the program ( Optional )

Output:

```
x_reg value before set operation is: 92 10010010
x_reg value after set operation is: ba 10111010
x_reg value before clear operation is: 92 10010010
x_reg value after clear operation is: 10 00010000
x_reg value before toggle operation is: 92 10010010
x_reg value after toggle operation is: 83 10000011
```

2) Given the following register:

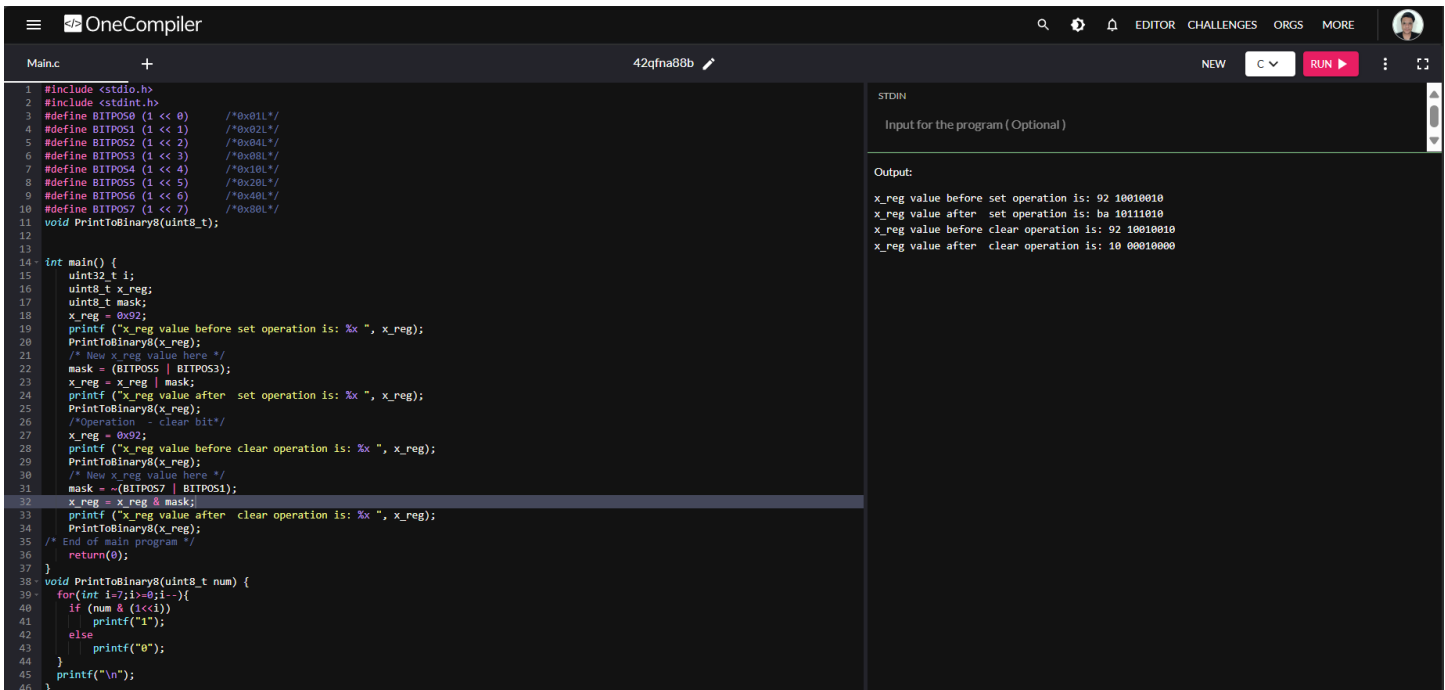
x_reg	1	0	0	1	0	0	1	0
Bit position	7	6	5	4	3	2	1	0

Write the code to perform the following operations **creating masks using #define**

a) Assume x\_reg = 92, bits 5 and 3 must be set (1) without modifying the others

b) Assume x\_reg = 92, bits 7 and 1 must be clear (0) without modifying the others

3) Report the final code and an image of your results



```
1 #include <stdio.h>
2 #include <stdint.h>
3 #define BITPOS0 (1 << 0) /*0x01*/
4 #define BITPOS1 (1 << 1) /*0x02*/
5 #define BITPOS2 (1 << 2) /*0x04*/
6 #define BITPOS3 (1 << 3) /*0x08*/
7 #define BITPOS4 (1 << 4) /*0x10*/
8 #define BITPOS5 (1 << 5) /*0x20*/
9 #define BITPOS6 (1 << 6) /*0x40*/
10 #define BITPOS7 (1 << 7) /*0x80*/
11 void PrintToBinary8(uint8_t);
12
13
14 int main() {
15     uint32_t i;
16     uint8_t x_reg;
17     uint8_t mask;
18     x_reg = 0x92;
19     printf("x_reg value before set operation is: %x ", x_reg);
20     PrintToBinary8(x_reg);
21     /* New x_reg value here */
22     mask = (BITPOS5 | BITPOS3);
23     x_reg = x_reg | mask;
24     printf("x_reg value after set operation is: %x ", x_reg);
25     PrintToBinary8(x_reg);
26     /*Operation - clear bit*/
27     x_reg = 0x92;
28     printf("x_reg value before clear operation is: %x ", x_reg);
29     PrintToBinary8(x_reg);
30     /* New x_reg value here */
31     mask = ~(BITPOS7 | BITPOS1);
32     x_reg = x_reg & mask;
33     printf("x_reg value after clear operation is: %x ", x_reg);
34     PrintToBinary8(x_reg);
35     /* End of main program */
36     return(0);
37 }
38 void PrintToBinary8(uint8_t num) {
39     for(int i=7;i>=0;i--){
40         if (num & (1<<i))
41             printf("1");
42         else
43             printf("0");
44     }
45     printf("\n");
46 }
```

Output:

```
x_reg value before set operation is: 92 10010010
x_reg value after set operation is: ba 10111010
x_reg value before clear operation is: 92 10010010
x_reg value after clear operation is: 10 00010000
```

2) Given the time value is packed as illustrated below:

31	18	17	12	11	6	5	0
Unused			Hours		Minutes		Seconds

```
uint32_t Time = 0x00009285;
uint32_t Minutes;
```

- What hour does 0x00009285 represent?
- Write code to extract the minutes
- Write code to insert the new hour. The new hour is 0x0B (11 in decimal)

3) Report the final code and an image of your results

The screenshot shows the OneCompiler IDE with a C program that manipulates a 32-bit time value. The code defines functions to print binary representations of 8-bit, 16-bit, and 32-bit integers. In the main function, it initializes a time value of 0x00009285 (5 hours, 6 minutes, 5 seconds). It then extracts the minutes (6) and inserts a new hour (0x0B or 11). The final output shows the updated time values.

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4
5 void PrintToBinary8(uint8_t i);
6 void PrintToBinary16(uint16_t i);
7 void PrintToBinary32(uint32_t i);
8 int main() {
9     uint32_t i;
10
11     uint32_t time = 0x00009285, new_time = 0x00; /* hour: 5, minutes:6, seconds: 5 */
12     uint32_t minutes = 0, hours = 0, new_hour;
13     uint32_t mask;
14     /* extracting minutes */
15     printf("Current time is: %x\n", time);
16     /* code here */
17     mask = (0x3f);
18     hours = (time >> 12) & mask;
19     printf("Hours are: %x\n", hours);
20     minutes = (time >> 6) & mask;
21     printf("Minutes are: %d\n", minutes);
22     new_hour = 0x0B;
23     new_time = time & ~(0x3f << 12);
24     new_time = time | (new_hour << 12);
25     printf("New hour is: %d\n", new_hour);
26
27     /* End of main program */
28     return(0);
29 }
30
31 void PrintToBinary8(uint8_t num) {
32     for(int i=7; i>=0; i--) {
33         if (num & (1 << i))
34             printf("1");
35         else
36             printf("0");
37     }
38     printf("\n");
39 }
40
41 void PrintToBinary16(uint16_t num) {
42     for(int i=15; i>=0; i--) {
43         if (num & (1 << i))
44             printf("1");
45         else
46             printf("0");
47     }
48     printf("\n");
49 }
50
51 void PrintToBinary32(uint32_t num) {
52     for(int i=31; i>=0; i--) {
53         if (num & (1 << i))
54             printf("1");
55         else
56             printf("0");
57     }
58     printf("\n");
59 }
```

Output:

```
Current time is: 9285
Hours are: 5
Minutes are: 6
New hour is: 11
```

1) Capture the following code:

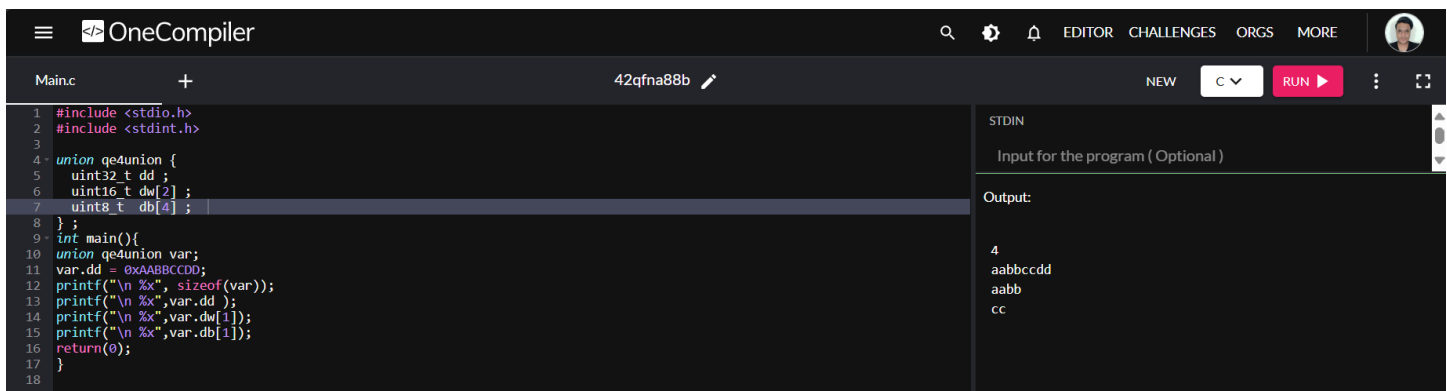
```
#include <stdio.h>
#include <stdint.h>

union ge4union {
    uint32_t dd ;
    uint16_t dw[2] ;
    uint8_t db[4] ;
};

int main(){
    union ge4union var;
    var.dd = 0xAABBCCDD;
    printf("\n %x", sizeof(var));
    printf("\n %x", var.dd );
    printf("\n %x", var.dw[1]);
    printf("\n %x", var.db[1]);
    return(0);
}
```

2) Analyze the code and indicate the expected result

3) Report the results



The screenshot shows the OneCompiler online IDE interface. The editor displays the C code from the previous block. The output panel on the right shows the results of running the program. The output is as follows:

```
STDIN
Input for the program ( Optional )

Output:

4
aabbccdd
aabb
cc
```

2) Repeat the same exercise covered in the previous quick experiment.

31	18	17	12	11	6	5	0
Unused		Hours		Minutes		Seconds	

uint32\_t Minutes;

a) Write the code to initialize the value as illustrated below:

timevar.reg = 0x00009285;

b) Write code to extract the minutes and print the value

c) Write code to insert the new hour. The new hour is 0x0A (10 in decimal) and print the new hour.

```

1 #include <stdio.h>
2 #include <stdint.h>
3
4 uint8_t main(void){
5     typedef union{
6         uint32_t reg;
7         struct{
8             uint32_t  seconds :6,
9                     minutes :6,
10                    hours   :6,
11                    :14;
12        }bits;
13    }MY_TIME;
14    MY_TIME time;
15    uint32_t seconds = 0, minutes = 0, hours = 0, new_hour = 0;
16    time.reg = 0x00009285;
17    printf("Current time = %x\n", time.reg);
18    seconds = time.bits.seconds;
19    minutes = time.bits.minutes;
20    hours = time.bits.hours;
21    printf("Seconds = %d\n", seconds);
22    printf("Minutes = %d\n", minutes);
23    printf("Hours = %d\n", hours);
24    new_hour = 0x0A;
25    time.bits.hours = new_hour;
26    printf("New Hour = %d", time.bits.hours);
27    return 0;
28 }
  
```

Output:

```

Current time = 9285
Seconds = 5
Minutes = 10
Hours = 9
New Hour = 10
  
```