QE1 (Boolean and bitwise operations)

1) Capture and run the following code

```c
#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 && (!3) || 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

EDITOR CHALLENGES ORGS MORE

Main.c 42qfna88b NEW C RUN

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


void PrintToBinary8(uint8_t);


int8_t main() {
    uint32_t i;
    uint8_t x_reg;
    /* Boolean and bit wise operations */
    /* Class example */
    uint8_t a = 5 || (!3) && 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");
}
```

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

8 && (!10) || 14

8 && (0) || 14

0 || 14

1

8 & (~10) | 14

8 & (~1010) | 14

8 & (0101) | 14

1000 & (0101) | 14

0000 | 1110

1110 = E

OneCompiler

EDITOR CHALLENGES ORGS MORE

Main.c 42qfna88b NEW C RUN

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


void PrintToBinary8(uint8_t);


int8_t main() {
    uint32_t i;
    uint8_t x_reg;
    /* Boolean and bit wise operations */
    /* Class example */
    //uint8_t a = 5 || (!3) && 6;
    //uint8_t b = (5 | (~3)) & 6;
    int a = 8 && (!10) || 14;
    int b = 8 & (~10) | 14;

    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");
}
```

STDIN

Input for the program (Optional)

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

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

c) 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

EDITOR CHALLENGES ORGS MORE

Main.c 42qfna88b NEW C RUN

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


void PrintToBinary8(uint8_t);


int main() {
    uint32_t i;
    uint8_t x_reg;
    uint8_t mask;

    x_reg = 0x92;
    printf ("x_reg value before set operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /* New x_reg value here */
    mask = (1<<5 | 1<<3);
    x_reg = x_reg | mask;
    printf ("x_reg value after  set operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /*Operation - clear bit*/
    x_reg = 0x92;
    printf ("x_reg value before clear operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /* New x_reg value here */
    mask = ~(1<<7 | 1<<1);
    x_reg = x_reg & mask;
    printf ("x_reg value after  clear operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /*Operation  - toggle bit*/
    x_reg = 0x92;
    printf ("x_reg value before toggle operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /* New x_reg value here */
    mask = (1<<4 | 1<<0);
    x_reg = x_reg ^ mask;
    printf ("x_reg value after  toggle operation is: %x ", x_reg);
    PrintToBinary8(x_reg);


/* 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");
}
```

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:

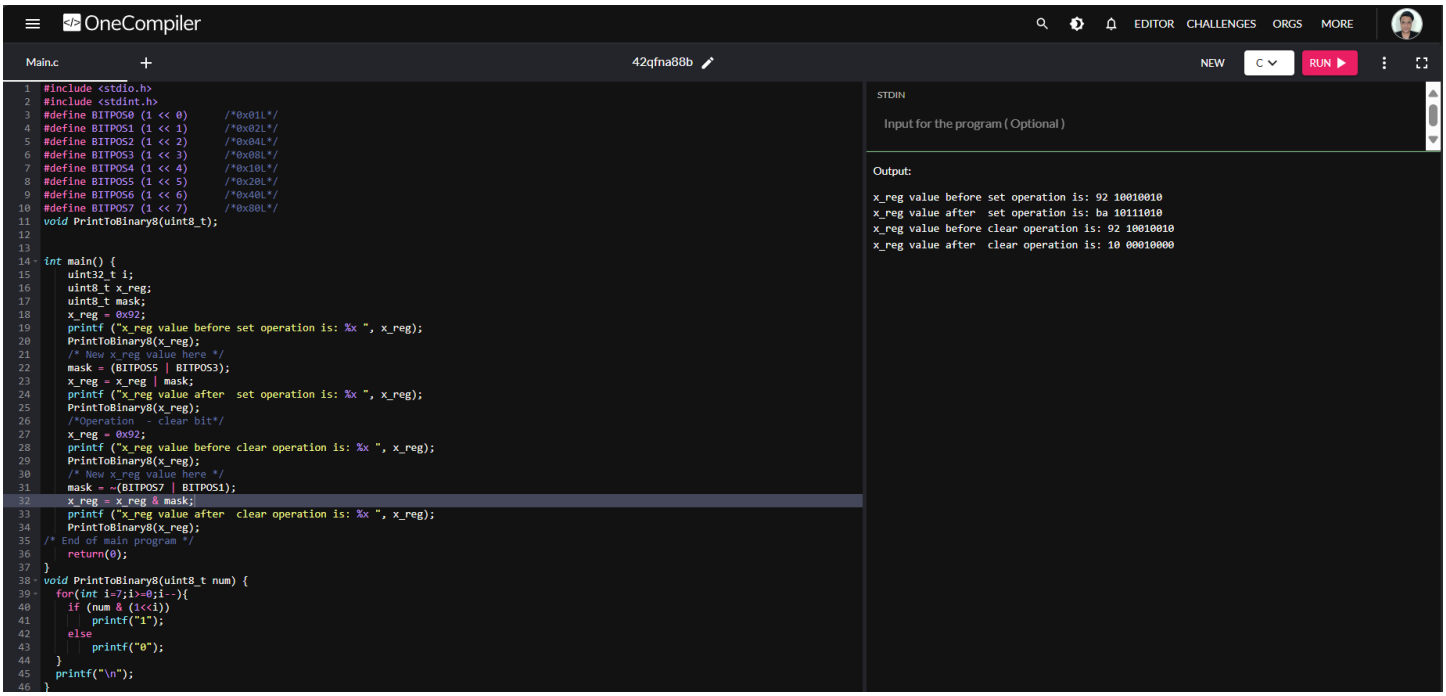
x_reg	1	0	0	1	0	0	1	0
Bit position	7	6	5	4	3	2	1	0

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




```c
#include <stdio.h>
#include <stdint.h>
#define BITPOS0 (1 << 0)      /*0x01L*/
#define BITPOS1 (1 << 1)      /*0x02L*/
#define BITPOS2 (1 << 2)      /*0x04L*/
#define BITPOS3 (1 << 3)      /*0x08L*/
#define BITPOS4 (1 << 4)      /*0x10L*/
#define BITPOS5 (1 << 5)      /*0x20L*/
#define BITPOS6 (1 << 6)      /*0x40L*/
#define BITPOS7 (1 << 7)      /*0x80L*/
void PrintToBinary8(uint8_t);


int main() {
    uint32_t i;
    uint8_t x_reg;
    uint8_t mask;
    x_reg = 0x92;
    printf ("x_reg value before set operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /* New x_reg value here */
    mask = (BITPOS5 | BITPOS3);
    x_reg = x_reg | mask;
    printf ("x_reg value after  set operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /*Operation  - clear bit*/
    x_reg = 0x92;
    printf ("x_reg value before clear operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
    /* New x_reg value here */
    mask = ~(BITPOS7 | BITPOS1);
    x_reg = x_reg & mask;
    printf ("x_reg value after  clear operation is: %x ", x_reg);
    PrintToBinary8(x_reg);
/* 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");
}
```

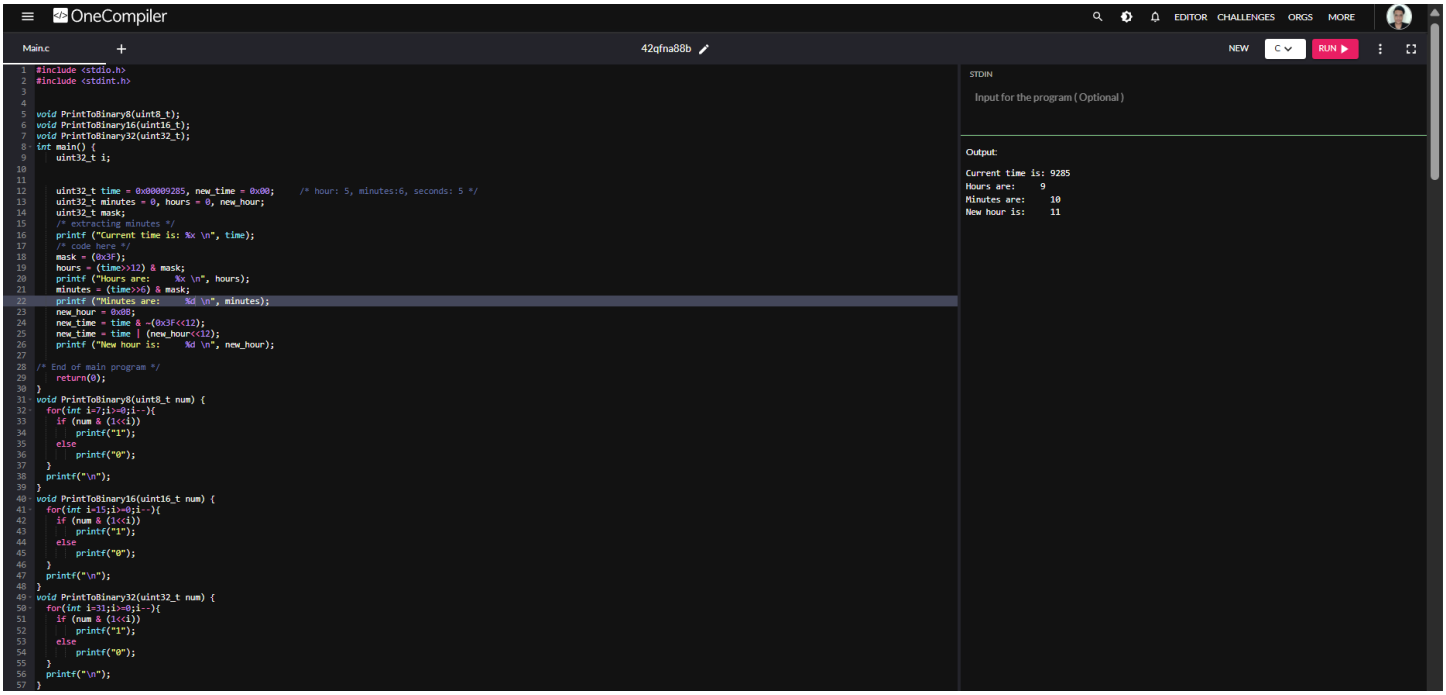
2) Given the time value is packed as illustrated below:

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

uint32_t Time = 0x00009285;
uint32_t Minutes;

a) What hour does 0x00009285 represent?
b) Write code to extract the minutes
c) 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




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


void PrintToBinary8(uint8_t);
void PrintToBinary16(uint16_t);
void PrintToBinary32(uint32_t);
int main() {
    uint32_t i;


    uint32_t time = 0x00009285, new_time = 0x00;     /* hour: 5, minutes:6, seconds: 5 */
    uint32_t minutes = 0, hours = 0, new_hour;
    uint32_t mask;
    /* extracting minutes */
    printf ("Current time is: %x \n", time);
    /* code here */
    mask = (0x3F);
    hours = (time>>12) & mask;
    printf ("Hours are:     %x \n", hours);
    minutes = (time>>6) & mask;
    printf ("Minutes are:     %d \n", minutes);
    new_hour = 0x0B;
    new_time = time & ~(0x3F<<12);
    new_time = time | (new_hour<<12);
    printf ("New hour is:     %d \n", new_hour);

/* 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");
}
void PrintToBinary16(uint16_t num) {
  for(int i=15;i>=0;i--){
    if (num & (1<<i))
        printf("1");
    else
        printf("0");
  }
  printf("\n");
}
void PrintToBinary32(uint32_t num) {
  for(int i=31;i>=0;i--){
    if (num & (1<<i))
        printf("1");
    else
        printf("0");
  }
  printf("\n");
}
```

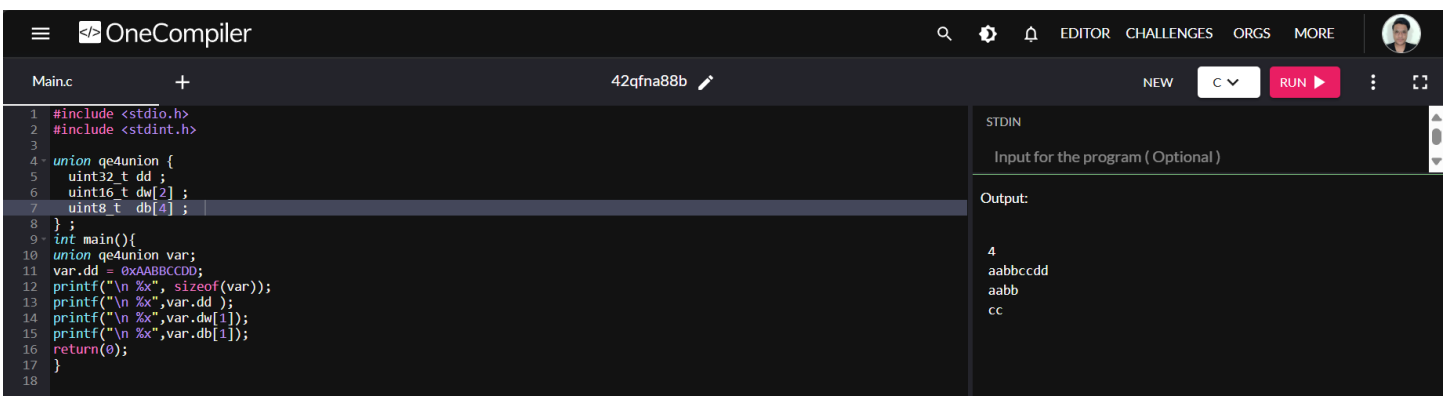
1) Capture the following code:

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

union qe4union {
  uint32_t dd ;
  uint16_t dw[2] ;
  uint8_t  db[4] ;
} ;
int main(){
union qe4union 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



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.

OneCompiler EDITOR CHALLENGES ORGS MORE

Main.c 42qfna88b NEW C RUN

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

uint8_t main(void){
  typedef union{
    uint32_t reg;
    struct{
      uint32_t  seconds :6,
                minutes :6,
                hours   :6,
                        :14;
    }bits;
  }MY_TIME;
  MY_TIME time;
  uint32_t seconds = 0, minutes = 0, hours = 0, new_hour = 0;
  time.reg = 0x00009285;
  printf("Current time = %x\n", time.reg);
  seconds = time.bits.seconds;
  minutes = time.bits.minutes;
  hours = time.bits.hours;
  printf("Seconds = %d\n", seconds);
  printf("Minutes = %d\n", minutes);
  printf("Hours = %d\n", hours);
  new_hour = 0x0A;
  time.bits.hours = new_hour;
  printf("New Hour = %d", time.bits.hours);
  return 0;
}
```

STDIN

Input for the program (Optional)

Output:

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