**Union**  
**Writer:** Sayak Haldar

**1) Size of a union is determined by size of the**

a)First member in the union

b)Last member in the union

c)Biggest member in the union

d)Sum of the sizes of all members

**Answer)** c) Biggest member in the union

**2)Comment on the following union declaration?**

1.#include <stdio.h>

2.union temp

3.{

4. int a;

5. float b;

6. char c;

7.};

union temp s = {1,2.5,’A'}; //REF LINE

Which member of the union will be active after REF LINE?

a)a

b)b

c)c

d)Such declarations are illegal

2)Answer is a) a

**Explanation:** union temp s = {1,2.5,’A'};

Here, {} works as an operator and choose the leftmost value among the values separated by comma to write in the allocated memory for union.

As a result,

#include <stdio.h>

union temp

{

char c;

float b;

int a;

};

int main()

{

union temp s={'A',2.5,1};

printf("%f %d %c\n",s.b,s.a,s.c);

return 0;

}

This will print 0.000000 65 A

Because, they are all printing the same memory location which contains value of ‘A’ which is 65.

However, reading the location as float will print 0.000000  
  
Why?

Now, the memory allocation for float will play a big part in it.

Now, consider it is as a Big Endian machine.   
  
The value stored is:  
  
00000000 00000000 00000000 01000001

**Now, the rules:**

Now, Memory allocation for float variable is the following:

**1st bit** : signed bit

**Next 8 bit** : exponent bit

**Next 23 bit** : value bit/significand bit

**Now, the exponent bit representation rule is the following:**

Exponents can be positive or negative, but instead of reserving another sign bit, they're encoded such that 10000000 represents 0, so 00000000 represents -128 and 11111111 represents 127.

**The significand bit rules are following:**

Each bit represents a negative power of 2 counting from the left, so:

01101 = 0 \* 2-1 + 1 \* 2-2 + 1 \* 2-3 + 0 \* 2-4 + 1 \* 2-5

= 0.25 + 0.125 + 0.03125

= 0.40625

**Now, using the rules, you can exactly calculate the value of**

00000000 00000000 00000000 01000001

So, exponent part is 00000000. Hence, -128  
  
Now, value part/significand bit part value is:

0000000 00000000 01000001

=0\*2-1+0\*2-2+0\*2-3+0\*2-4+0\*2-5+0\*2-6+0\*2-7+0\*2-8+0\*2-9+0\*2-10+0\*2-11+0\*2-12+0\*2-13+0\*2-14+0\*2-15+0\*2-16+1\*2-17+…….+1\*2-23

The final result would be (2-17+2-23)\*2-128

**3)What would be the size of the following union declaration?**

1.#include <stdio.h>

2.union uTemp

3.{

4. double a;

5. int b[10];

6. char c;

7.}u;

(Assuming size of double = 8, size of int = 4, size of char = 1)

a) 4

b)8

c)40

d)80  
  
**Answer is c) 40**

Explanation: size of union=Biggest member in the union. Here, biggest member of the union is the array b of type int

**4)What type of data is held by variable u int this C code?**

1.#include <stdio.h>

2.union u\_tag

3.{

4. int ival;

5. float fval;

6. char \*sval;

7.} u;

**The variable u here**

a)Will be large enough to hold the largest of the three types;

b)Will be large enough to hold the smallest of the three types;

c)Will be large enough to hold the all of the three types

d)None of the mentioned

**Answer) a)**

**5) Members of a union are accessed as\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.**

a)union-name.member

b)union-pointer->member

c)Both a & b

d)None of the mentioned

**Answer is c) Both a & b**

Depending upon whether we declare a variable of an union or a pointer to an union.

**6)What is the output of this C code?**

1.#include <stdio.h>

2.struct

3.{

4. char \*name;

5. union

6. {

7. char \*sval;

8. } u;

9.} symtab[10];

The first character of the string sval by either of

a)\*symtab[i].u.sval

b)symtab[i].u.sval[0]

c)You cannot have union inside structure

d)Both a & b

**Answer is d) Both a & b**

Explanation: I guess no explanation is needed for symtab[i].u.sval[0]

But, a little explanation is needed for \*symtab[i].u.sval \*symtab[i].u.sval

.this is Element selection by reference operator Whereas,

\* this is Indirection (Reference) operator

**Now, '.' operator has higher precedence than '\*' operator**

**'.' operator has associativity from left to right and '\*' operator has associativity from right to left**

\*symtab[i].u.sval

\*(symtab[i].u.sval)

symtab[I].u.sval[0] (Derived by pointer arithmatic)

**7)What is the output of this C code(size of int and float is 4)?**

1.#include <stdio.h>

2.union

3.{

4. int ival;

5. float fval;

6.} u;

7.int main()

8.{

9. printf("%d", sizeof(u));

10. return 0;

11.}

a)16

b)8

c)4

d)32

**Answer) is c) 4**We skip the optional union-tag part here. We declare a variable u of the union

**8)What is the output of this C code?**

1.#include <stdio.h>

2.union stu

3.{

4. int ival;

5. float fval;

6.};

7.int main()

8.{

9. union stu r;

10. r.ival = 5;

11. printf("%d", r.ival);

12. return 0;

13.}

a)9

b)Compile time error

c)16

d)5

**Answer is d) 5**

**9)What is the output of this C code?**

1.#include <stdio.h>

2.union

3.{

4. int x;

5. char y;

6.}p;

7.int main()

8.{

9. p.x = 10;

10. printf("%d\n", sizeof(p));

11. return 0;

12.}

a)Compile time error

b)sizeof(int) + sizeof(char)

c)Depends on the compiler

d)sizeof(int)

Answer) d) sizeof(int)

**10)What is the output of this C code?**

1.#include <stdio.h>

2.union

3.{

1. int x;
2. char y;

6.}p;

1. int main()
2. {
3. p.y = 60;
4. printf("%d\n", sizeof(p));
5. }

a)Compile time error

b)sizeof(int) + sizeof(char)

c)Depends on the compiler

d)sizeof(char)

**Answer is c) Depends on the compiler**

Explanation: In case of some compiler. The size allocated for the union=the biggest member of the union (Always). In case of of other compilers, the size allocated for the union=the member you are trying to make active explicitly. Like, in this case, the program is trying to make member 'y' of type type char active explicitly.

**11)What is the output of this C code?**

1.#include <stdio.h>

2.union p

3.{

4. int x;

5. char y;

6.};

7.int main()

8.{

9. union p p, b;

10. p.y = 60;

11. b.x = 12;

12. printf("%d\n", p.y);

13. return 0;

14.}

a)Compile time error

b)Depends on the compiler

c)60

d)Undefined behaviour

**Answer is c) 60**

Unlike the previous question, it does not compiler. Because, even if the memory is allocated for a character, we can print a character’s value with %d. It will print the ASCII value of it.

**12)What is the output of this C code?**

1.#include <stdio.h>

2.union p

3.{

4. int x;

5. char y;

6.}k = {1, 97};

7.int main()

8.{

9. printf("%d\n", k.y);

10. return 0;

11.}

a) Compile time error

b)97

c)a

d)1

**Answer) is d) 1**

**13)What is the output of this C code?**

1.#include <stdio.h>

2.union p

3.{

4. int x;

5. char y;

6.}k = {.y = 97};

7.int main()

8.{

9. printf("%d\n", k.y);

10. return 0;

11.}

a)Compile time error

b)97

c)a

d)Depends on the standard

**Answer is b) 97**

**14)What is the output of this C code?**

1.#include <stdio.h>

2.union p

3.{

4. int x;

5. float y;

6.};

7.int main()

8.{

9. union p p, b;

10. p.x = 10;

11. printf("%f\n", p.y);

12. return 0;

13.}

a)Compile time error

b)Implementation dependent

C) 10.000000

d)0.000000

**Answer) b) implementation dependent.**Floating datatype's memory allocation could play a big part here. (We already discussed it a number of times.

Check the question 2’s explanation.

**15) Which of the following share a similarity in syntax?**

1.Union, 2. Structure, 3. Arrays and 4. Pointers

a)3 and 4

b)1 and 2

c)1 and 3

d)1, 3 and 4

**Answer) 1 and 2**

**16)What is the output of this C code?**

1.#include <stdio.h>

2.union utemp

3.{

4. int a;

5. double b;

6. char c;

7.}u;

8.int main()

9.{

10. u.c = 'A';

11. u.a = 1;

12. printf("%d", sizeof(u));

13.}

The output will be: (Assuming size of char = 1, int = 4, double = 8)

a)1

b)4

c)8

d)13

**Answer) c) 8**

**17)What is the output of this C code?**

1.#include <stdio.h>

2.union utemp

3.{

4. int a;

5. int b;

6.}u;

7.int main()

8.{

9. u.a = 97;

10. u.b = 88;

11. printf("u.a=%d",u.b=%d\n”,u.a,u.b);

12. return 0;

13.}

The output will be: (Assuming size of char = 1, int = 4, double = 8)

a)97 88

b)97 97

c)88 88

d)compilation error

**Answer) c) 88 88**

**Explanation:** The union is only as big as necessary to hold its largest data member. The other data members are allocated in the same bytes as part of that largest member. The details of that allocation are implementation-defined, **and it's undefined behavior to read from the member of the union that wasn't most recently written.**

However, in most of the cases, the value allocated to the biggest member is the value shown when we are trying the read other members, if other members are big enough to hold the value.

So, the same could give undefined behaviour.

**18)What is the output of this C code?**

#include <stdio.h>

2.union utemp

3.{

4. int a;

5. char b;

6.}u;

7.int main()

8.{

9. u.a = 97;

10. u.b = 'B';

11. printf("%d", sizeof(u));

12. return 0;

13.}

a)1

b)4

c)5

d)None of the following

**Answer is b) 4**