DATE: LAB REPORT NO.: **8** SET: **B**
TITLE OF THE PROGRAM: **Union**

OBJECTIVES

- Understand the concept of unions in C programming.
- Learn how to define and use unions.
- Observe the behavior of union members when different values are assigned.
- Analyze the memory efficiency provided by unions.

REQUIREMENTS

1. C Compiler (e.g., GCC)
2. Computer System
3. IDE or Text Editor
4. OS compatible with the software

THEORY

A union is a special data type in C that allows storing different data types in the same memory location. While a union can have many members, only one member can hold a value at any given time. This feature makes unions an efficient way of using memory when the data stored is mutually exclusive.

Syntax
The syntax for defining a union is as follows:

```
union [union name] {
    member definition;
    member definition;
    ...
    member definition;
} var1, var2;
```

PROCEDURE (Program Code, Comment, and Output)

1. C Program to Implement Union

Program Code:
```
#include <stdio.h>
```

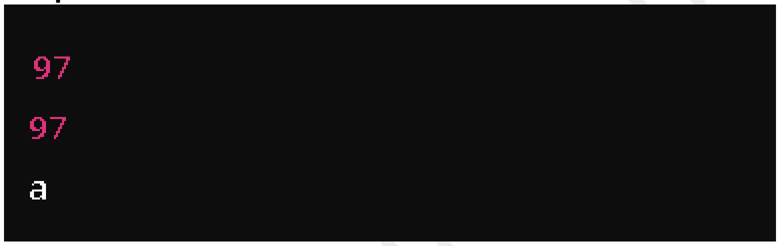
```
union item {
   int x;
   int y;
   char ch;
};

int main()
 {
   union item it;
   it.x = 12;
   it.y = 47;
   it.ch = 'a';

   printf("%d\n", it.x);
   printf("%d\n", it.y);
   printf("%c\n", it.ch);

   return 0;
}
```

Output



```
97
97
a
```

Explanation:

- A union **item** is defined with three members: **int x**, **int y**, and **char ch**.
- In the main function, a variable **it** of type **union item** is declared.
- The members of the union are assigned values sequentially: **it.x = 12**, **it.y = 47**, and **it.ch = 'a'**.
- The printf statements then print the values of **it.x**, **it.y**, and **it.ch.**

CONCLUSION

The lab demonstrates that only one member of a union can hold a value at any given time. When a new value is assigned to a different member, the previous value is overwritten. This behavior is due to all members sharing the same memory location, making unions an efficient way to manage memory for mutually exclusive data.

By understanding unions, we can better utilize memory in scenarios where different data types are used interchangeably.