

WEEK-END ASSIGNMENT-09

Structure and Union

Operating Systems Workshop (CSE 3541)

Problem Statement:

Experiment with user-defined data types to store heterogeneous data and their processing.

Assignment Objectives:

To learn about structure & union data types and their implications in programming.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realise the outcome of that question in relevant to systems programming. You may use additional pages on requirement.

Programming/ Output Based Questions:

1. Select the invalid member of the following structure;

```
struct oswcourse{  
    int secid;  
    float avgm;  
    char present;  
    int *marks();  
    int teacher();  
}o1,o2;
```

Output with explanation

2. Detect any invalid member present in the given structure;

```
struct date{  
    int m,d,y;  
};  
struct stud{  
    char name[20];  
    struct stud *p;  
    struct date *d;  
    int (*)fun(int, int);  
};
```

Output with explanation

3. The following structure template is allowed or not in ANSI C.

```
struct person{  
    int a;  
    struct health{  
        int a;  
    }h;  
};
```

Output with explanation

4. The following declaration is correct or wrong.

```
struct person{
    int a;
    union health{
        int w;
    }h;
};
```

Output with explanation

5. The following declaration is correct or wrong.

```
union person{
    int a;
    struct health{
        int e;
    }h;
};
```

Output with explanation

6. Check the declaration of the structure. Write a valid conclusion whether **Line-5** can be valid member or not.

```
struct person{
    int ht;
    float wt;
    char color;
    struct person p; /*Line- 5 */
};
```

Output with explanation

7. Write valid or invalid form of the followings.

- (1) union{....}u;
- (2) union u{.....};
- (3) struct{.....}s;
- (4) struct s{.....};

Output with explanation

8. Decide the output of the code snippet;

```
int main(){
    struct student{
        int h;
        int w;
        int m;
    };
    struct student s1={20,40,50};
    struct student *ptr=&s1;
    printf("%d\n",*((int *)ptr+2));
    return 0;
}
```

Output and reason▼

9. Find the output of the code snippet;

```
struct s{int *p;};
int main(){int a=200;struct s s1;
    s1.p=&a;    *(s1.p)=*(s1.p)+100;
    printf("%d %d\n",a,*(s1.p));
    return 0;}
```

Output and reason▼

10. Draw the node connectivity of the structure **s1** and determine the output of the code snippet that simulates the array of structures and also the self-referential structure;

```
int main() {
    struct s1{
        char *z;
        int i;
        struct s1 *p;
    };
    struct s1 a[]={{"SOA",1,a+1},
                   {"ITER",4,a+2},
                   {"CSE",5,a}
    };
    struct s1 *ptr=a;
    printf("%s%s%s\n",a[0].z,a[1].z,a[2].z);
    printf("%s%s%s", (*ptr).z, ptr->z,a[2].p->z);
    return 0;
}
```

Draw figure and Output▼

11. Draw the node connectivity of the structure **s1** and determine the output of the code snippet that simulates the array of structures and also the self-referential structure;

```
int main() {
    struct s1{
        char *z;
        int i;
        struct s1 *p;
    };
    struct s1 a[]={{"SOA",1,a+1},
                   {"ITER",2,a+2},
                   {"CSE",3,a}};
    struct s1 *ptr=a;
    printf("%s\n", ++(ptr->z));
    printf("%s\n", a[(++ptr->i).z]);
    printf("%s\n", a[--(ptr->p->i)].z);
    printf("%d\n",--a[2].i);
    return 0;
}
```

Draw figure and Output▼

12. An initialization of array of structures given in the following code snippet. Find the output with pointer manipulation and operator precedence rules.

```
int main() {
    struct test{
        int i;
        char *c;
    };
    struct test st[]={5, "Cse-Engg",
                      4, "computer",
                      6, "Electrical",
                      8, "Mechanical",
                      7, "All-Engg"
    };
    struct test *p=st;
    printf("%s\n", ++(p++->c));
    printf("%c\n", *p++->c);
    printf("%d\n", ++p->i);
}
```

```
printf("%s\n",p[0].c);
printf("%s\n",p->c);
return 0;}
```

Output▼

13. Conclude the output of the code snippet based on pointer and operator precedence on a nested structure case.

```
int main(){
    struct out{
        char ch[10];
        char *str;
    };
    struct b{
        char *c;
        struct out o;
    };
    struct b s2={"ODISHA", "KHURDA", "JOYDEV"};
    printf("%s %s %s\n", s2.c, s2.o.str, s2.o.ch);
    printf("%s %s\n", ++s2.c, ++s2.o.str);
    return 0;
}
```

Output and reason▼

14. Find the output of the code snippet;

```
int main(){
    union unit{
        int marks;
        int roll;
    }s1,s2;
    s2.roll=23;
    s1.marks=60;
    printf("%d..%d\n", s1.marks, s2.roll);
    return 0;
}
```

Output and reason▼

15. Find the output of the code snippet;

```
int main(){
    union unit{
        int marks;
        int roll;
    }s1,s2;
    s2.roll=23;
    s2.marks=60;
    printf("Check memory alloc for union\n");
    printf("%d..%d\n", s2.marks, s2.roll);
    return 0;
}
```

Output and reason▼

16. Declare two variable of the structure type **planet_t**

```
typedef struct{
    char name[30];
    double diameter;
    int moons;
    double or_time,ro_time;
}planet_t;
```

Declaration▼

17. Initialize one of the variable of the question-16 structure with values "jupiter", 142.34, 16, 11.9, 9.23;

Initialization▼

18. Declare a pointer to the structure type **planet_t** and initialize the structure components with the help of the pointer.

Initialization▼

19. Assuming a 24-hr clock and the structure type **time_t** is defined below.

```
typedef struct{ int hour, minute, second; };
```

Write a program that contains a function **new_time(...)** that returns as its value an updated time based the original time of day and the number of seconds that have elapsed since the previous update time. if time now were 21-58-32 and elapsed second 97, the original time now is 22-00-09. The function prototype to compute the new time is **time_t new_time(time_t time_of_day, int elapsed_secs);**

Code here▼

Code here▼

20. Define a structure type **auto_t** to represent an automobile. Include components for the make and model (strings), the odometer reading, the manufacture and purchase dates (use another user-defined type called **date_t**), and the gas tank (use a user-defined type **tank_t** with components for tank capacity and current fuel level, giving both in gallons). Write I/O functions **scan_date**, **scan_tank**, **scan_auto**, **print_date**, **print_tank**, and **print_auto**, and also write a driver function that repeatedly fills and displays an auto structure variable until **EOF** is encountered in the input file. Here is a small data set to try:

```
Mercury Sable 99842 1 18 2001 5 30 1991 16 12.5
Mazda Nava jo 123961 2 20 1993 6 15 1993 19.3 16.7
```

Code here▼

Code here▼

Code here▼

21. Numeric addresses for computers on the international network Internet are composed of four parts, separated by periods, of the form

xx.yy.zz.mm

where **xx**, **yy**, **zz**, and **mm** are positive integers. Locally, computers are usually known by a nickname as well. You are designing a program to process a list of Internet addresses, identifying all pairs of computers from the same locality. Create a structure type called **address_t** with components for the four integers of an Internet address and a fifth component in which to store an associated nickname of ten characters. Your program should read a list of up to 100 addresses and nicknames terminated by a sentinel address of all zeros and a sentinel nickname.

Sample Data
111.22.3.44 platte
555.66.7.88 wabash
111.22.5.66 green
0.0.0.0 none

The program should display a list of messages identifying each pair of computers from the same locality, that is, each pair of computers with matching values in the first two components of the address. In the messages, the computers should be identified by their nicknames.

Example Message
Machines platte and green are on the same local
network.

Follow the messages by a display of the full list of addresses and nicknames. Include in your program a **scan_address** function, a **print_address** function, and a **local_address** function. Function **local_address** should take two address structures as input parameters and return 1 (for true) if the addresses are on the same local network, and 0 (for false) otherwise.

Code here▼

Code here▼