**Union assignments**

Mandatory

1. Refer the code below and comment on size of the given structure considering

a. Structure as union

b. Structure as struct

c. arr

d. uarr

\_\_\_ Job

{

char name[32];

unsigned short ucount;

float salary;

int workerNo;

char \*orgname;

};

\_\_\_ Job myvar; //could of union or of struct

Struct Job arr[10];

Union Job uarr[10];

**A. Structure Size**

**(a) Structure as union**

**Size: 32 bytes**

**(b) Structure as struct**

**Size: 48 bytes**

**(c) arr**

**Size: 480 bytes**

**(d) uarr**

**Size: 320 bytes**

2. Refer Job datastructure in Q#1 above. Using uarr, perform below operations.

a. Read and store salary

b. Read and store workerNo

Comment on values of output if salary and workerNo are printed in order. Justify your statement.

1. **Using uarr to Read/Store Salary and Worker Number**

**Code:**

uarr[0].salary = 5000.50;

uarr[0].workerNo = 42;

printf("Salary: %.2f\n", uarr[0].salary);

printf("Worker No: %d\n", uarr[0].workerNo);

**Comment:**

* **Output:**  
  Salary will be undefined/corrupted.  
  Worker No will be 42.
* **Justification:**  
  Writing workerNo overwrites salary due to shared memory in the union.

3. Refer Job datastructure in Q#1 above. Assume that myvar is a structure variable. If I need to place 2 bytes (i.e 0x0102) as ucount using a char \*ptr then list all possible statements that can be used in \_\_\_\_\_.

[Let solutions include cases such as

i. using base address of ucount

ii. using relative address of ucount w.r.t to base address of myvar]

int main()

{

char \*ptr = &myvar;

\_\_\_\_\_\_\_\_\_\_\_ = 0x01;

\_\_\_\_\_\_\_\_\_\_\_ = 0x02

}

1. **Storing ucount Using a Pointer**

**Code:**

**(i) Using base address of ucount:**

char \*ptr = (char \*)&myvar.ucount;

\*ptr = 0x01;

\*(ptr + 1) = 0x02;

**(ii) Using relative address of ucount w.r.t myvar:**

char \*ptr = (char \*)&myvar;

\*(ptr + offsetof(struct Job, ucount)) = 0x01;

\*(ptr + offsetof(struct Job, ucount) + 1) = 0x02;

4o