WEEK 12 ASSIGNMENT SOLUTION

1. Which of the following are themselves a collection of different data types?
    a) String
    b) Array
    c) Character
    d) Structure

Solution: (d) Structure

Structure is a user defined data type available in C that allows combining data items of different kinds.

2. Which of the following comments about the usage structures is true?

    a) Storage class can be assigned to individual member
    b) Individual members can be initialized within a structure type declaration
    c) The scope of the member name is confined to the particular structure, within which it is defined
    d) None

Solution: (c) The scope of the member name is confined to the particular structure, within which it is defined

3. What will be output of the program?

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int i;
   int *ptr = (int *) malloc(5 * sizeof(int));

   for (i=0; i<5; i++)
      *(ptr + i) = i;
   printf("%d ", *ptr++);
   printf("%d ", (*ptr)++);
   printf("%d ", *ptr);
   printf("%d ", *++ptr);
   printf("%d ", ++*ptr);
  return 0;
}
```

a) Error

b) 0 1 2 3 4

c) 1 2 3 4 5

d) 0 1 2 2 3

Solution: (d) 0 1 2 2 3

The important things to remember for handling such questions are--

Prefix ++ and * operators have the same precedence and right to left associativity. Postfix ++ has higher precedence than the above two mentioned operators and associativity is from left to right.
We can apply the above two rules to guess all

*ptr++ is treated as *(ptr++)
*++ptr is treated as *(++ptr)

++*ptr is treated as ++(*ptr)

4. What will be output of the program?

```
#include <stdio.h>
int fun(int arr[]) {
  arr = arr+1;
  printf("%d ", arr[0]);
}
int main() {
  int arr[3] = {5, 10, 15};
  fun(arr);
  printf("%d ", arr[0]);
  printf("%d ", arr[1]);
  return 0;
}
```

a) 5 10 10

b) 10 5 15

c) 10 5 10

d) 10 15 5

Solution: (c) 10 5 10

In C, array parameters are treated as pointers So the variable *arr* represents an array in main(), but a pointer in fun().

5. What is the output of the following C code? Assume that the address of x is 2000 (in decimal) and an integer requires four bytes of memory

```
#include <stdio.h>
int main()
{
  unsigned int x[4][3] = {{1, 2, 3}, {4, 5, 6},
                          {7, 8, 9}, {10, 11, 12}};
```

```
    printf("%u, %u, %u", x+3, *(x+3), *(x+2)+3);
}
```

a) 2036 2036 2036
b) 2012 4 2204
c) 2036 10 10
d) 2012 4 6

Solution: (a) 2036 2036 2036

x = 2000

Since x is considered as a pointer to an array of 3 integers and an integer takes 4 bytes, value of x + 3 = 2000 + 3*3*4 = 2036

The expression, *(x + 3) also prints the same address as x is 2D array.
The expression *(x + 2) + 3 = 2000 + 2*3*4 + 3*4 = 2036

6. The program will allocate …….bytes to ptr. Assume sizeof(int)=4.

```
#include<stdio.h>
#include<stdlib.h>
 int main()
 {
    int *ptr;
    ptr = (int*)malloc(sizeof(int)*4);
    ptr = realloc(ptr,sizeof(int)*2);
    return 0;
 }
```

a) 2
b) 4
c) 8
d) None

Solution: (c) 8

We can also use the realloc() to change memory block size.

7. What is the output of the following C program?

```
#include <stdio.h>
int main()
{
int *p, a=10;
p=&10;
printf("%d",*p);
return 0;
}
```

a) 10
b) a
c) address of a
d) compilation error

Solution: (d) A pointer variable can be assigned as the address of any constant. Thus, the compiler will show error as “[Error] lvalue required as unary '&' operand”.

8. What is the output?

```
#include<stdio.h>
int main()
{
struct xyz{ int a;};
struct xyz obj1={11};
struct xyz obj2 = obj1;
printf("%d", obj2.a);
obj2.a = 101;
printf("%d", obj1.a);
printf("%d", obj2.a);
return 0;
}
```

a) 1111011
b) 1111101
c) 1110111
d) 1110011

Solution: (b) 1111101
Initially, obj2 stores the value of obj1, which is 11. When obj2 is changed, obj1 remains the same and therefore, it prints 1111101

9. Calling a function f with a an array variable a[3] where a is an array, is equivalent to
   a) f(a[3])
   b) f(*(a + 3))
   c) f(3[a])
   d) all of the mentioned

Solution: (d) all the methods are correct.

WEEK 12 ASSIGNMENT SOLUTION

10. What is the output of the following C program?

```
#include <stdio.h>
struct p
{
   int x;
   char y;
};

int main()
{
   struct p p1[] = {1,21,69,42,64};
   struct p *ptr1 = p1;
   int x = (sizeof(p1) / 4);
   if (x == sizeof(int) + 2*sizeof(char))
      printf("True");
   else
      printf("False");
   return 0;
}
```

a) True
b) False
c) No output
d) Compilation error

Solution: (a) True
Due to padding operations of structures the sizeof struct p1 is 24.
The reason is as follows:
The memory assignment of struct p is as follows:

int 1st byte	int 2nd byte	int 3rd byte	int 4th byte	char

To store the second element of p1 i.e. 21, 3 bytes are padded, which makes it 8 bytes.
While storing the 3rd element, the memory gets allocated for 8*2=16 bytes as shown below.

	1st element				2nd element		
	3rd element				Blank spaces		

Finally the memory structure of p1 will look like this

	1st element				2nd element		
	3rd element				4th element		
	5th element				Blank spaces		

In the program, x = 24/4 = 6. And sizeof(int) +2* sizeof(char) is also 6.
Therefore, the TRUE is printed.