1D, 2D, MultiDimensional Array Assignments:

Mandatory

1D Array

1. Refer the code snippet and answer the queries

```
int main()
{
   int array[100];
   int *ptr;
   // do something
}
```

Q1: Can pointer be used in Array-style syntax? e.g. ptr[10], ptr[0]
A: Yes, the pointer can be used in Array style syntax

Q2: Can Array be used in Pointer-style syntax? e.g. *array, *(array + 0), *(array + 10)
A: Yes, an array can be used in pointer style syntax

Q3: is ptr++ valid?
A: Yes, it is valid. It moves to the next address

Q4: is array++ valid?
A: It is invalid. Array address cannot be changed

Q5: what is sizeof(array)?
A: Sizeof(array) gives the total size of array

Q6: what is sizeof(ptr)?
A: Gives the sizeof ptr generally 4 bytes

2. Refer the code snippet below. Comment on the other elements (other than those that are explicitly initialized) of all array variables in code snippet below.

```
#define MAX  100
int main()
{
        int arr[MAX] = {11,22,33};
        int arr1[MAX]={0};
        static int arr2[MAX];
}
```

A. Arr[3] to arr[99] → initialized to 0
Arr[0] to arr[99] → initialized to 0
All elements are initialized to 0 as they are static

3. Refer the program “array_pointer.c”. Add a function getmax() to find the maximum in the array and call in main() and display the result.

A. getMax() function:

```
 printf("\n");
 max=getMax(numbers);
 printf("\n Maximum Number: %d\n",max);

 return 0;
}
int getMax(int n[])
{
    int Max=0,i;
    for(i=0;i<MAX;i++)
    {
        if(*(n+i)>Max)
            Max=*(n+i);
}
return Max;
}
```

```
user46@trainux01:~/ass$ ./a.out
44      22      11      55      33
44      22      0       55      33

 Maximum Number: 55
```

4. Extend the code given below to read N and a start value from the user to perform the given operations.

#define MAX 100

```
int main()
{
        int arr[MAX] = {11,22,33};
}
```

Add the following functions choosing proper input, output and return.

a. init() - Use the inputs to initialize the first N elements of the array with N consequetive values starting with given start value .
b. update() – increment value of every element in the array
c. display() – display the contents of array

```c
#include<stdio.h>
#define MAX 100
void init(int [],int,int);
void update(int [],int);
void display(int [],int);
int main()
{
    int arr[MAX]={11,22,33};
    int N,s;

    printf("Enter n value :\n ");
    scanf("%d",&N);
    printf("Enter start value :\n");
    scanf("%d",&s);

    init(arr,N,s);
    display(arr,N);
    printf("\n\n");
    update(arr,N);
    display(arr,N);
    return 0;
}

void init(int arr[],int N,int s)
{
    int i;
    for(i=0;i<N;i++)
        arr[i]=s+i;
}
void update(int  arr[],int N)
{
    int i;
    for(i=0;i<N;i++)
    arr[i]++;
}
void display(int arr[],int N)
{
    int i;
    for(i=0;i<N;i++)
        printf("%d\t", arr[i]);
}
```

```
Enter n value :
 5
Enter start value :
10
10      11      12      13      14

11      12      13      14      15
```

2D, MultiDimensional Arrays

1. Implement sort() to sort a given array. Refer the code snippet below.

```
int main()
{
        char arr[]= “xaybz”;

        sort(arr, sizeof(arr)/sizeof(arr[0]);
        return 0;
}
```

```c
#include<stdio.h>
void sort(char [],int);

int main()
{
    char arr[]="xaybz";
    int n = sizeof(arr)/sizeof(arr[0])-1;
    sort(arr,n);
    printf("Sorted Array : %s\n", arr);

}
void sort(char arr[],int n)
{
    int i,j;
    char t;
    for(i=0;i<n-1;i++)
    {
     for(j=0;j<n-i-1;j++)
     {
        if(arr[j]>arr[j+1])
        {
        t=arr[j];
        arr[j]=arr[j+1];
        arr[j+1]=t;
    }
}

}
}
```

```
Sorted Array : abxyz
```

2. Refer the code snippet below.

```
int main()
{
        char arr[][3] = {
        sort(arr, sizeof(arr)/sizeof(arr[0]);
        return 0;
}
```

Allow user to perform the following operations.

a. init() - initialize the array and return 0
b. search_update() – search for a given element in array and if found update it to given value and return 0 else return 1
c. display() – traverse and display array contents

For the functions, pass array and other required arguments to functions and return as per requirement

```c
int main() {
    int arr[ROWS][COLS] = {{0}};
    int c, search, new;
    while (1) {
        printf("\nChoose an operation:\n");
        printf("1. Initialize Array\n");
        printf("2. Search and Update Element\n");
        printf("3. Display \n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
       switch (c) {
            case 1:
                if (init(arr) == 0) {
                    printf("Array initialized successfully.\n");
                }
                break;
            case 2:
                printf("Enter the element to search and update: ");
                if (scanf("%d", &search) != 1) {
                    printf(" enter an integer.\n");
                    while (getchar() != '\n');
                    break;
                }
                printf("Enter the new value to update: ");
                if (scanf("%d", &new) != 1) {
                    printf(" enter an integer.\n");
                    while (getchar() != '\n');
                    break;
                }
                search_update(arr, search, new);
                break;
            case 3:
                display(arr);
                break;
            case 4:
                printf("Exiting program.\n");
                return 0;
            default:
                printf("Enter valid number\n");
        }
    }

    return 0;
INSERT
```

```c
int init(int arr[ROWS][COLS]) {
    printf("Enter values for the array \n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("arr[%d][%d]: ", i, j);
            if (scanf("%d", &arr[i][j]) != 1) {
                printf("Invalid input! Please enter integers.\n");
                return 1;              }
        }
    }
    return 0;
}

int search_update(int arr[ROWS][COLS], int search, int new) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            if (arr[i][j] == search) {
                arr[i][j] = new;
                return 0;
            }
        }
    }
    printf("Element %d not found in the array.\n", search);
    return 1;
}

void display(int arr[ROWS][COLS]) {
    printf("\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
}
```

```
Choose an operation:
1. Initialize Array
2. Search and Update Element
3. Display
4. Exit
Enter your choice: 1
Enter values for the array
arr[0][0]: 1
arr[0][1]: 2
arr[0][2]: 4
arr[1][0]: 6
arr[1][1]: 3
arr[1][2]: 5
arr[2][0]: 7
arr[2][1]: 8
arr[2][2]: 9
Array initialized successfully.

Choose an operation:
1. Initialize Array
2. Search and Update Element
3. Display
4. Exit
Enter your choice: 3

1 2 4
6 3 5
7 8 9

Choose an operation:
1. Initialize Array
2. Search and Update Element
3. Display
4. Exit
Enter your choice: 2
Enter the element to search and update: 2
Enter the new value to update: 6
```

```
Enter your choice: 2
Enter the element to search and update: 2
Enter the new value to update: 6

Choose an operation:
1. Initialize Array
2. Search and Update Element
3. Display
4. Exit
Enter your choice: 3

1 6 4
6 3 5
7 8 9

Choose an operation:
1. Initialize Array
2. Search and Update Element
3. Display
4. Exit
Enter your choice: 4
Exiting program.
```