Objective:

- To get a grip on Array Structure manipulation on heap
- Finally, Get your hands dirty with pointers ☺
- Dynamic Memory Allocation

Given to PF – SE-F17

Task-1: *Sets*
Write the following functions to support the set operations.

Each set will be represented by three identifiers that is:

- int *
    - It will point to an array of integers on heap treated as set of integers
- int
    - it will store size of array pointed by '*int **' treated as size of set/array
- int
    - it will store number of Elements stored in set

Functions:

1. void createSet (int * * set, int n);
2. bool addElement (int * set, int * noe, int capacity, int element);
3. bool removeElement (int * set, int * noe, int capacity, int element);
4. bool searchElement (int * set, int noe, int element);
5. int searchElementPosition (int * set, int noe, int element);
6. bool isEmpty(int noe);
7. bool isFull(int noe, int capacity);
8. void displaySet (int * set, int noe);
9. int* calcIntersection (int * setA, int * setB, int setANoe, int setBNoe, int * newSetNoe, int * newSetCapcity);
10. int isSubset (int * setA, int * setB, int setANoe, int setBNoe);
    10.1. return 1 if proper subset
    10.2. return 2 if improper subset
    10.3. return 0 if not a subset
11. void reSize(int ** setA, int * setANoe, int *setACapacity, int newSize);
12. void displayPowerSet (int * set , int noe);
13. void creatClone (int * sourceSet, int sourceNoe, int sourceCapacity, int * * targetSet, int * targetNoe, int * targetCapacity);
14. deallocateSet (int * * set);

Sample Run:

```
int setACapacity = 10;
int setANOE  = 0
int * setA

int setBCapacity = 7;
int setBNOE  = 0
int * setB;

createSet ( &setA, setACapacity );

createSet ( &setB, setBCapacity );

addElement ( setA, & setANOE, setACapacity, 5 );
addElement ( setA, & setANOE, setACapacity, 15 );
addElement ( setA, & setANOE, setACapacity, 9 );
addElement ( setA, & setANOE, setACapacity, 10 );

cout<<"Set A Elements : ";
displaySet ( setA , setANOE );

addElement ( setB, & setBNOE, setACapacity, 9 );
addElement ( setB, & setBNOE, setACapacity, 17 );
addElement ( setB, & setBNOE, setACapacity, 95 );

cout<<"Set B Elements : ";
displaySet ( setB , setBNOE );


int setCCapacity = 0;
int setCNOE  = 0
int * setC;

setC = intersection ( setA, setB, setANOE, setBNOE, & SetCNoe, & setCCapacity);

cout<<"Set C Elements : ";
displaySet ( setC , setCNOE );

cout<<"\nPower Set of B : ";
displayPowerSet ( setB , setBNOE );
```

Console

```
Set A Elements : { 5, 15, 9, 10 }
Set B Elements : { 9, 17, 95 }
Set C Elements : { 9 }

Power Set of B : { {}, { 9 }, { 17 }, { 95 }, { 9, 17 }. { 9, 95 }, { 17, 95 }, { 9, 17, 95 } }
```