Lab 04
OOP – BCS

Task 1: Write following functions to:

a. display one dimensional array in tabular form using mapping function:

```
void display1D_2D(int *x, const int ROWS, const int COLS)
```

Example:

```
Consider array values 23 46 78 90 42 36 12 38 14 48 52 58

display1D_2D( array, 4, 3)

      23    46    78
      90    42    36
      12    38    14
      48    52    58

display1D_2D( array, 2, 6)

      23    46    78    90    42    36
      12    38    14    48    52    58
```

b. copy one dimensional array (source) into two dimensional array (target), given size of one dimensional array is equal to the multiple of rows & columns in two dimensional array:

```
void convert1D_2D(int *src, const int SIZE, int **tar, const int ROWS, const int COLS)
```

Use following code for testing:

```
void show(int *x, const int SIZE){
      for (int i = 0 ; i < SIZE; i++)
            cout << x[i] << ' ';
      cout << "\n\n";
}
void show(int **x, const int ROWS, const int COLS){
      int i, j;
      for (i = 0 ; i < ROWS; i++){
            for (j = 0 ; j < COLS; j++)
                  cout << x[i][j] << ' ';
            cout << '\n';
      }
      cout << '\n';
}
int main(){
      int i, j;
      int x[] = {23,46,78,90,42,36,12,38,14,48,52,58};
      show (x, 12);
      int **y=new int*[3];
      for (i=0;i<3;i++)
            y[i]=new int[4];
      int **z=new int*[4];
      for (i=0;i<4;i++)
            z[i]=new int[3];
      convert1D_2D(x, 12, y, 3, 4);
      show (y, 3, 4);
      convert1D_2D(x, 12, z, 4, 3);
      show (z, 4, 3);
      return 0;
```

c. copy two dimensional array (source) into one dimensional array (target), given size of one dimensional array is equal to the multiple of rows & columns in two dimensional array:

```
void convert2D_1D(int **src, const int ROWS, const int COLS, int *tar, const int SIZE)
```

Use following code for testing:

```
void show(int *x, const int SIZE){
	for (int i = 0 ; i < SIZE; i++)
		cout << x[i] << ' ';
	cout << "\n\n";
}
void show(int **x, const int ROWS, const int COLS){
	int i, j;
	for (i = 0 ; i < ROWS; i++){
		for (j = 0 ; j < COLS; j++)
			cout << x[i][j] << ' ';
		cout << '\n';
	}
	cout << '\n';
}
int main(){
	int i,j;
	int x[] = {23,46,78,90,42,36,12,38,14,48,52,58};
	show (x, 12);
	int **y=new int*[3];
	for (i=0;i<3;i++)
		y[i]=new int[4];
	int z[12];
	convert1D_2D(x, 12, y, 3, 4);//Function created in part (b)
	show (y, 3, 4);
	convert2D_1D(y, 3, 4, z, 12);
	show (z, 12);
	return 0;
}
```

Task 2: Write following functions to:

a. return dynamic array having common elements of array 1 and array 2 and store count of common elements in last parameter commonCount:

```
int* findCommon(int *a1, const int SIZE1, int *a2, const int SIZE2, int &commonCount)
```

Use following code for testing:

```
void show(int *x, const int SIZE){
	for (int i = 0 ; i < SIZE; i++)
		cout << x[i] << ' ';
	cout << "\n";
}
int main(){
	int x1[] = {23,46,78,90,42};
	int x2[] = {36,12,38,14,48,52,58};
	show (x1, 5);
	show (x2, 7);
	int commonCount;
	int *common=findCommon(x1, 5, x2, 7, commonCount);
	show (common, commonCount);
	delete []common;
	return 0;
}
```

return dynamic array having common elements of array 1 and array 2 and store count of common elements in last parameter so that user can get number of elements in common array:

b. remove extra spaces from array. There should be one space in between words, and after full stop. Lastly, there shouldn't be any extra spaces at the start or at the end, see example given at the end:

```
void removeExtraSpacesInTarget (char *array)
```

String before function calling: `This is heavy. You  are busy.  I am going  .`
String after function calling: `This is heavy. You are busy. I am going.`

c. place elements of source 2D array (stored in 1D array) into target array at given location. Fourth & fifth parameter (r, c) is the position to place elements of source array; whereas; second & third parameter (tRows, tCols) are number of rows and columns in target array. Lastly, seventh & eighth parameters (SRows, sCols) are number of rows and columns in source array. Assume that source array has rows and columns always less than equal to rows and columns of target array and target array has sufficient size to place all elements of source array into target array. Read function parameters and match code to understand parameters order and purpose. Also see the code and output to understand the functionality of the function:

```
void placeAt (int *tar,int tRows,int tCols,int r, int c, int *src, int sRows, int sCols)
```

```
int x[] = {23,46,78,89,42,36,12,38,14,48,52,58,39,29,49,59,67,83,94,96,18,29,19,99};
show (x, 4,6);
int y[] = {10,20,30,40,50,60,70,80,90};
placeAt (x,4,6,1,1,y,3,3);
show (x, 4,6);
```

```
23 46 78 89 42 36
12 38 14 48 52 58
39 29 49 59 67 83
94 96 18 29 19 99

23 46 78 89 42 36
12 10 20 30 52 58
39 40 50 60 67 83
94 70 80 90 19 99
```

```
-------------
int x[] = {23,46,78,89,42,36,12,38,14,48,52,58,39,29,49,59,67,83,94,96};
show (x, 4, 5);
int z[] = {-1,-2,-3,-4};
placeAt (x,4,6,1,2,z,2,2);
show (x, 4, 5);
```

```
23 46 78 89 42
36 12 38 14 48
52 58 39 29 49
59 67 83 94 96

23 46 78 89 42
36 12 -1 -2 48
52 58 -3 -4 49
59 67 83 94 96
```

********* END OF LAB (Best of Luck) ********