

## **Problem Statement:**

### **Part(A) (2):**

The following function uses arrays:

```
void calcSalary() {
    int salary[3];
    for (int i=0;i<3;++i) {
        cout <<"Enter Salary: ";
        cin >>salary[i]; }
    cout << endl;
    cout << "Entered salaries are: " << endl;
    for (int i=0; i<3; ++i) {
        cout << salary[i] << " ";
    }
    cout << endl;
    cout << "Updated salaries are: " << endl;
    for (int i=0; i<3; ++i) {
        salary[i] = salary[i] + salary[i]/(i+1);
        cout << salary[i] << " ";
    }
}
```

Create a new function `calcSalaryArrayByPointerArithmetic`. This new function should modify the above function `calcSalary` as follows: The new function should use a pointer to the array `salary`, and pointer arithmetic to access and assign values to the array in all the for loops (see slides on “Pointers and Arrays” for reference).

### **Part(B)- (3):**

*(Dice Rolling)* Write a program that simulates the rolling of two dice. The program should use `rand` to roll the first die and should use `rand` again to roll the second die. The sum of the two values should then be calculated. [Note: Each die can show an integer value from 1 to 6, so the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 being the least frequent sums.] Figure 7.26 shows the 36 possible combinations of the two dice. Your program should roll the two dice 36,000 times. Use a one-dimensional array to tally the numbers of times each possible sum appears. Print the results in a tabular format. Also, determine if the totals are reasonable (i.e., there are six ways to roll a 7, so approximately one-sixth of all the rolls should be 7).

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

**Fig. 7.26** | The 36 possible outcomes of rolling two dice.

### **Part(C)- (2):**

- Implement a **generic** class to represent a dynamic two-dimensional array. Number of rows and columns should be private data members. [3]
- The constructor should take value of rows and columns as its parameters and create dynamic memory. [2]
- Implement a function input() that takes values in the array from console [2]

Create two objects of this class in main function; one for 2D dynamic array of integers and other of characters. [3]

### **Part(D)- (2):**

```
void process(char [ ], const int size);
```

```
void main(){  
char x[10];  
cin>>x;  
process(x,10); }
```

Modify the above code for the following scenario:

We want the user to enter only numbers in this array. If all the values in the array are numbers then the array **x** should be sent to function **process(...)** for further processing.

If user enters any non-numeric value in this array then it is an exception and **x** should not be sent to **process()** instead the code should throw an error message **"Array cannot be processed further"**.

You may use following built-in function to check whether a given character is digit or not. It returns true if c is a digit

```
bool isdigit( char c ) ;
```

### **Non-Coding-Part (1 Mark):**

Create a **text** file named `reflect.txt` that contains your **detailed** description of the topics that you have learned in completing this Lab and mention any issues that caused you difficulty and how you solved them.

