Date:__ / __ / ____

Experiment 1

1. a. Write a C program to implement an array of fixed dimension. Perform the following operations inserting an element into an array and display the contents of an array. Note: user input is required to enter the size of an array.

Program

```
#include <stdio.h>
int main()
{
   int n;
   printf("Enter the size of the array:");
   scanf("%d",&n);
   int arr[n];

   printf("Enter the %d values to store it in array: \n", n);
   for(int i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
   printf("The values stored in the array are: \n");
   for(int i=0;i<n;i++)
   {
      printf("%d \n",arr[i]);
   }

}
```

Sample Output

```
Enter the size of the array: 4
Enter the 4 values to store it in array: 1
2
3
4
The values stored in the array are:
1
2
3
4
```

1. b. Evaluate the Number Plates of Vehicles.

Write a C program to insert a vehicle registration number into an array of 1x10 array. The objective is to apply strict conventional rules of vehicle number plate registration as follows: The first two positions of array must be filled with state name, next two positions must be filled with district code, followed by next two positions with serial number of characters and penultimate with a number of a vehicle. KA-09-MN-3865. The program needs to verify the registration of a vehicle in the same format and print the registration of a vehicle and "Accept" else print "Reject" with a proper message.

Rules to check.

i. First two positions of an array must be a State Code.
ii. Next two positions of an array must be District Code.
iii. Next two Positions of an array must be a Serial Number of an RTO.
iv. Penultimate r positions of an array must be a vehicle number.

Program

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
// Function to verify if a vehicle registration number is valid
bool verifyRegistration(char registration[])
{
   // Check if the registration number has exactly 10 characters
   if (strlen(registration) != 10)
   {
      return false;
   }

   // Check if the first two characters are alphabets (e.g., "KA" for Karnataka)
   for (int i = 0; i < 2; i++)
   {
      if (!isalpha(registration[i]))
      {
         return false;
      }
   }

   // Check if the next two characters are numbers (e.g., "09" for a specific district)
   for (int i = 2; i < 4; i++)
   {
      if (!isdigit(registration[i]))
      {
         return false;
      }
   }

   // Check if the next two characters are alphabets (e.g., "MN" for serial number)
   for (int i = 4; i < 6; i++)
```

```
    {
        if (!isalpha(registration[i]))
        {
            return false;
        }
    }

    // Check if the penultimate two characters are numbers (e.g., "3865" for vehicle number)
    for (int i = 6; i < 10; i++)
    {
        if (!isdigit(registration[i]))
        {
            return false;
        }
    }

    return true;
}

int main() {
    char registration[11];
    printf("Enter a vehicle registration number: ");
    scanf("%s", registration);

    if (verifyRegistration(registration))
    {
        printf("Accept: Vehicle registration number is valid.\n");
    }
    else
    {
        printf("Reject: Vehicle registration number is invalid.\n");
    }

    return 0;
}
```

Sample Output

Enter a vehicle registration number: KA09MN3865
Accept: Vehicle registration number is valid.

Enter a vehicle registration number: KAABMN3865
Reject: Vehicle registration number is invalid.

Date:__ / __ / ____

Experiment 2

2. a. Write a C program that multiplies two matrices, ensuring that the number of columns in the first matrix is equal to the number of rows in the second matrix. Display the resulting matrix. Note: Prompt user to enter the size of the matrix.

Program

```
#include <stdio.h>
int main() {
   int m, n, p, q;

   printf("Enter the number of rows and columns of the first matrix: ");
   scanf("%d %d", &m, &n);

   printf("Enter the number of rows and columns of the second matrix: ");
   scanf("%d %d", &p, &q);

   if (n != p) {
      printf("Matrix multiplication is not possible. Column of the first matrix must be equal to
the row of the second matrix.\n");
      return 1;
   }

   int firstMatrix[m][n], secondMatrix[p][q], resultMatrix[m][q];

   printf("Enter elements of the first matrix:\n");
   for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
         scanf("%d", &firstMatrix[i][j]);
      }
   }

   printf("Enter elements of the second matrix:\n");
   for (int i = 0; i < p; i++) {
      for (int j = 0; j < q; j++) {
         scanf("%d", &secondMatrix[i][j]);
      }
   }

   // Matrix multiplication
   for (int i = 0; i < m; i++) {
      for (int j = 0; j < q; j++) {
         resultMatrix[i][j] = 0;
         for (int k = 0; k < n; k++) {
```

```
                resultMatrix[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
            }
        }
    }

    printf("Resultant matrix after multiplication:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < q; j++) {
            printf("%d ", resultMatrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

2. b. Validate the sudoku board.
Write a C program to determine the validity of a 9x9 Sudoku board. The task was to apply strict Sudoku rules to only the filled cells. With lines of code, scan the rows, columns, and 3x3 sub grids for reptations. The program verifies the Sudoku board's integrity and print "**Accept**" if it satisfies all the sudoku board condition, else print "**Reject**".

Conditions to check
i. Every row should encompass the numbers 1 through 9, ensuring no duplication.
ii. Each column should consist of the numbers 1 to 9, ensuring no duplication.
iii. Within the nine 3x3 sub-boxes of the grid, no repetitions allowed.

Sample Input
//copy and paste the input
```
int sudoku[9][9] = {
        {5, 3, 0, 0, 7, 0, 0, 0, 0},
        {6, 0, 0, 1, 9, 5, 0, 0, 0},
        {0, 9, 8, 0, 0, 0, 0, 6, 0},
        {8, 0, 0, 0, 6, 0, 0, 0, 3},
        {4, 0, 0, 8, 0, 3, 0, 0, 1},
        {7, 0, 0, 0, 2, 0, 0, 0, 6},
        {0, 6, 0, 0, 0, 0, 2, 8, 0},
        {0, 0, 0, 4, 1, 9, 0, 0, 5},
        {0, 0, 0, 0, 8, 0, 0, 7, 9}
    };
```

Sample Output
Accept

```
#include <stdio.h>
int isValidSudoku(int board[9][9]) {
    // Check rows
    for (int i = 0; i < 9; i++) {
        int row[10] = {0};
        for (int j = 0; j < 9; j++) {
            if (board[i][j] != 0 && row[board[i][j]] == 1) {
                return 0; // Invalid Sudoku
            }
            row[board[i][j]] = 1;
        }
    }

    // Check columns
    for (int j = 0; j < 9; j++) {
        int col[10] = {0};
        for (int i = 0; i < 9; i++) {
            if (board[i][j] != 0 && col[board[i][j]] == 1) {
                return 0; // Invalid Sudoku
            }
```

```
            col[board[i][j]] = 1;
        }
    }

    // Check 3x3 subgrids
    for (int block = 0; block < 9; block++) {
        int subgrid[10] = {0};
        for (int i = block / 3 * 3; i < block / 3 * 3 + 3; i++) {
            for (int j = block % 3 * 3; j < block % 3 * 3 + 3; j++) {
                if (board[i][j] != 0 && subgrid[board[i][j]] == 1) {
                    return 0; // Invalid Sudoku
                }
                subgrid[board[i][j]] = 1;
            }
        }
    }

    return 1; // Valid Sudoku
}

int main() {
    int sudoku[9][9] = {
        {5, 3, 0, 0, 7, 0, 0, 0, 0},
        {6, 0, 0, 1, 9, 5, 0, 0, 0},
        {0, 9, 8, 0, 0, 0, 0, 6, 0},
        {8, 0, 0, 0, 6, 0, 0, 0, 3},
        {4, 0, 0, 8, 0, 3, 0, 0, 1},
        {7, 0, 0, 0, 2, 0, 0, 0, 6},
        {0, 6, 0, 0, 0, 0, 2, 8, 0},
        {0, 0, 0, 4, 1, 9, 0, 0, 5},
        {0, 0, 0, 0, 8, 0, 0, 7, 9}
    };

    if (isValidSudoku(sudoku)) {
        printf("Valid Sudoku\n");
    } else {
        printf("Invalid Sudoku\n");
    }

    return 0;
}
```