Week 6

ANIRUDH ROWJEE `PES2UG20CS050`

Task 1

Write a program to generate Pascal's Triangle using two-dimensional arrays. Eg

```
1
1 1
1 2 1
1 3 3 1
```

`main.c`

```c
#include <stdio.h>
#include "utils.h"

int main()
{
    int limit;
    printf("Enter the depth of the Pascal Triangle > ");
    scanf("%d", &limit);

    int pascal[100][100];
    for (int i = 0; i < limit; i++)
    {
        for (int j = 0; j < limit; j++)
        {
            pascal[i][j] = 0;
        }
    }

    generatePascalTriangle(limit, pascal);
    for (int i = 0; i < limit; i++)
    {
        for (int j = 0; j < limit; j++)
        {
            if (pascal[i][j] == 0)
            {
                printf("");
            }
            else
            {
                printf("%d ", pascal[i][j]);
            }
        }
        printf("\n");
    }

    return 0;
}
```

`utils.h`

```
void generatePascalTriangle(int depth, int multi_array[100][100]);
```

`utils.c`

```
#include <stdio.h>
#include "utils.h"

void generatePascalTriangle(int depth, int multi_array[100][100])
{
    // function to generate a Pascal's Triangle in a
    // depthXdepth array

    // this is the row iterator
    for (int i = 0; i < depth; i++)
    {

        multi_array[i][0] = 1;
        // this is the inner iterator, start from 1 as the
        // 0th column will be one by default
        for (int j = 1; j <= i; j++)
        {
            // printf("%d %d\n", i, j);
            // take care of the diagonal
            if (i == j)
            {
                multi_array[i][j] = 1;
            }
            // strict branching to prevent rewrites
            else
            {
                int sum = multi_array[i-1][j-1] + multi_array[i-1][j];
                // printf("Added %d at (%d, %d)\n", sum, i, j);
                multi_array[i][j] = sum;
            }
        }
    }
}
```

Console

```
nidavellir :: UE20CS152/week5/t1 » ./main
Enter the depth of the Pascal Triangle > 4
1
1 1
1 2 1
1 3 3 1
nidavellir :: UE20CS152/week5/t1 » ./main
Enter the depth of the Pascal Triangle > 5
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
nidavellir :: UE20CS152/week5/t1 »
```

Task 2

Write a C program to read elements in a matrix and check whether the given matrix is symmetric matrix or not.

main.c

```c
#include <stdio.h>
#include <stdbool.h>
#include "utils.h"

int main()
{
    int order;
    printf("Enter the Order of the Matrix > ");
    scanf("%d", &order);

    // create the matrix
    int matrix[order][order];
    int transposed[order][order];

    // load user values - original matrix
    for (int i = 0; i < order; i++)
    {
        for (int j = 0; j < order; j++)
        {
            printf("Enter the Element at (%d, %d) > ", i+1, j+1);
            scanf("%d", &matrix[i][j]);
        }
    }

    // load zeroes - transposed matrix
    LoadZeroes(order, transposed);
    printf("\n");
    printf("The Matrix => \n");
    PrintMatrix(order, matrix);
    printf("\n");
    // transpose the first matrix
    TransposeSquareMatrix(order, matrix, transposed);
    printf("The Transpose => \n");
    PrintMatrix(order, transposed);
    printf("\n");
    // check for equality
    printf("The Matrix is %s\n",
            CheckSquareMatrixEquality(order, matrix, transposed)
                ? "Symmetric" : "Asymmetric"
          );
    return 0;
}
```

utils.h

```c

void LoadZeroes(int depth, int (*)[depth]);

void TransposeSquareMatrix(
        int depth, int (*)[depth], int (*)[depth]
        );
bool CheckSquareMatrixEquality(
        int depth, int (*)[depth], int(*)[depth]
        );

void PrintMatrix(int depth, int (*)[depth]);
```

utils.c

```c
#include <stdio.h>
#include <stdbool.h>
#include "utils.h"


void PrintMatrix(int depth, int (*matrix)[depth])
{
    // function to print matrix
    for (int i = 0; i < depth; i++)
    {
        for (int j = 0; j < depth; j++)
        {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void LoadZeroes(int depth, int (*matrix)[depth])
{
    // fills a matrix with zeroes
    for (int i = 0; i < depth; i++)
    {
        for (int j = 0; j < depth; j++)
        {
            matrix[i][j] = 0;
        }
    }
}

void TransposeSquareMatrix(
        int depth, int (*original)[depth], int (*transpose)[depth]
)
{
    // transpose the first matrix
    for (int i = 0; i < depth; i++)
    {
        for (int j = 0; j < depth; j++)
        {
            transpose[i][j] = original[j][i];
        }
    }
}

bool CheckSquareMatrixEquality(
        int depth, int (*first)[depth], int (*second)[depth]
)
{
    bool equal = true;
    for (int i = 0; i < depth; i++)
    {
        for (int j = 0; j < depth; j++)
        {
            if (first[i][j] != second[i][j])
            {
                equal = false;
                return equal;
            }
        }
    }
    return equal;
}
```

Console

```
nidavellir :: UE20CS152/week5/t2 » ./main
Enter the Order of the Matrix > 2
Enter the Element at (1, 1) > 1
Enter the Element at (1, 2) > 2
Enter the Element at (2, 1) > 3
Enter the Element at (2, 2) > 4

The Matrix =>
1 2
3 4

The Transpose =>
1 3
2 4

The Matrix is Asymmetric
nidavellir :: UE20CS152/week5/t2 » ./main
Enter the Order of the Matrix > 3
Enter the Element at (1, 1) > 1
Enter the Element at (1, 2) > 0
Enter the Element at (1, 3) > 0
Enter the Element at (2, 1) > 0
Enter the Element at (2, 2) > 1
Enter the Element at (2, 3) > 0
Enter the Element at (3, 1) > 0
Enter the Element at (3, 2) > 0
Enter the Element at (3, 3) > 1

The Matrix =>
1 0 0
0 1 0
0 0 1

The Transpose =>
1 0 0
0 1 0
0 0 1

The Matrix is Symmetric
nidavellir :: UE20CS152/week5/t2 »
```

Task 3

Write a C program to take in two dates from the user, and compare the two dates, providing the appropriate error messages.

`main.c`

```c
#include <stdio.h>
#include "utils.h"

int main()
{
    Date d1, d2;
    printf("Date 1 > \n");
    GetDate(&d1);
    printf("Date 2 > \n");
    GetDate(&d2);

    char* outcome;
    int comparison_result = CompareDates(d1, d2);

    switch (comparison_result)
    {
        case 1:
            outcome = "Date 2 Occurred After Date 1.";
            break;
        case 0:
            outcome = "Date 2 and Date 1 are the same.";
            break;
        case -1:
            outcome = "Date 2 Occurred Before Date 1.";
            break;
    };
    printf("%s\n", outcome);
    return 0;
}
```

`utils.h`

```

struct date {
    int day;
    int month;
    int year;
};
typedef struct date Date;
int CompareDates(Date date1, Date date2);
void GetDate(Date* date);
```

`utils.c`

```
#include <stdio.h>
#include "utils.h"


int mod(int quantity)
{
    if (quantity > 0)
    {
        return 1;
    }
    else if (quantity < 0)
    {
        return -1;
    }
    else
    {
        return 0;
    }
}

/*
 * This function tells us whether date2 is equal to, greater than
 * or lesser than date1
 * ----------------------------
```

```c
 * After => returns 1
 * On => returns 0
 * Before => returns -1
 */
int CompareDates(Date date1, Date date2)
{
    // compare two dates
    Date relativeDate = {
        mod(date2.year - date1.year),
        mod(date2.month - date1.month),
        mod(date2.day - date1.day),
    };

    switch (relativeDate.year)
    {
        case 1:
            return 1;
            break;
        case -1:
            return -1;
            break;
        case 0:
            switch (relativeDate.month)
            {
                case 1:
                    return 1;
                    break;
                case -1:
                    return -1;
                    break;
                case 0:
                    switch (relativeDate.day)
                    {
                    case 1:
                        return 1;
                        break;
                    case -1:
                        return -1;
                        break;
                    case 0:
                        return 0;
                        break;
                    }
            }
            break;
    }
    // we'll never reach here.
    return 10;
}


// function to get the date from the user as input
void GetDate(Date* date)
{
    printf("Enter the Date as DD/MM/YYYY >> ");
    scanf("%d/%d/%d", &date->day, &date->month, &date->year);
}
```

Console

```
nidavellir :: UE20CS152/week5/t3 » ./main
Date 1 >
```

```
Enter the Date as DD/MM/YYYY >> 1/1/2000
Date 2 >
Enter the Date as DD/MM/YYYY >> 2/1/2000
Date 2 Occurred After Date 1.
nidavellir :: UE20CS152/week5/t3 » ./main
Date 1 >
Enter the Date as DD/MM/YYYY >> 10/12/2002
Date 2 >
Enter the Date as DD/MM/YYYY >> 2/3/2002
Date 2 Occurred Before Date 1.
nidavellir :: UE20CS152/week5/t3 » ./main
Date 1 >
Enter the Date as DD/MM/YYYY >> 5/8/2021
Date 2 >
Enter the Date as DD/MM/YYYY >> 5/8/2021
Date 2 and Date 1 are the same.
nidavellir :: UE20CS152/week5/t3 »
```

Task 4

Write a C program to add and subtract two complex numbers by passing the appropriate structures to a function.

`main.c`

```c
#include <stdio.h>
#include "utils.h"

int main()
{
    complex_num c1, c2;

    printf("Enter the Details of Complex Number 1.\n");
    load_complex(&c1);
    printf("Enter the Details of Complex Number 2.\n");
    load_complex(&c2);

    printf("the sum is > ");
    display_complex(add(c1, c2));

    printf("the difference is > ");
    display_complex(subtract(c1, c2));

    return 0;
}
```

`utils.h`

```c
struct Complex
{
    float re;
    float im;
};

typedef struct Complex complex_num;

void display_complex(complex_num number);
void load_complex(complex_num* number);
```

```
complex_num add(complex_num c1, complex_num c2);
complex_num subtract(complex_num c1, complex_num c2);
```

`utils.c`

```
#include <stdio.h>
#include "utils.h"


void load_complex(complex_num* number)
{
    // function to accept a complex number from the user
    printf("Enter the Complex Number as 'A + Bi' >> ");
    scanf("%f + %fi", &number->re, &number->im);
}

void display_complex(complex_num number)
{
    printf("%f + %fi\n", number.re, number.im);
}

complex_num add(complex_num c1, complex_num c2)
{
    // adds two complex numbers
    complex_num c3;
    c3.re = c1.re + c2.re;
    c3.im = c1.im + c2.im;
    return c3;
}


complex_num subtract(complex_num c1, complex_num c2)
{
    // subtracts two complex numbers
    complex_num c3;
    c3.re = c1.re - c2.re;
    c3.im = c1.im - c2.im;
    return c3;
}
```

Console

```
nidavellir :: UE20CS152/week5/t4 » ./main
Enter the Details of Complex Number 1.
Enter the Complex Number as 'A + Bi' >> 3 + 4i
Enter the Details of Complex Number 2.
Enter the Complex Number as 'A + Bi' >> 1 + 2i
the sum is > 4.000000 + 6.000000i
the difference is > 2.000000 + 2.000000i
nidavellir :: UE20CS152/week5/t4 » ./main
Enter the Details of Complex Number 1.
Enter the Complex Number as 'A + Bi' >> 10 + 10i
Enter the Details of Complex Number 2.
Enter the Complex Number as 'A + Bi' >> 20 + 3i
the sum is > 30.000000 + 13.000000i
the difference is > -10.000000 + 7.000000i
nidavellir :: UE20CS152/week5/t4 »
```

Practice Program 1

Write a C program to fill a 5x5 Matrix as follows:

1. Populate the Upper-Left Diagonal with -1
2. Populate the Leading Diagaonal with Zeroes
3. Populate the Bottom-Right Diagonal with 1

`main.c`

```c
#include <stdio.h>
#include "utils.h"

int main()
{
    int limit;
    printf("enter the limit >> ");

    scanf("%d", &limit);
    int matrix[limit][limit];

    LoadZeroes(limit, matrix);
    FillMatrix(limit, matrix);
    PrintMatrix(limit, matrix);
    return 0;
}
```

`utils.h`

```c
void LoadZeroes(int depth, int (*)[depth]);
void PrintMatrix(int depth, int (*)[depth]);
void FillMatrix(int depth, int(*)[depth]);
```

`utils.c`

```c
#include <stdio.h>
#include <stdbool.h>
#include "utils.h"


void PrintMatrix(int depth, int (*matrix)[depth])
{
    // function to print matrix
    for (int i = 0; i < depth; i++)
    {
        for (int j = 0; j < depth; j++)
        {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void LoadZeroes(int depth, int (*matrix)[depth])
{
    // fills a matrix with zeroes
    for (int i = 0; i < depth; i++)
    {
        for (int j = 0; j < depth; j++)
        {
            matrix[i][j] = 0;
        }
    }
}

void FillMatrix(int depth, int (*matrix)[depth])
{
    // function to fill the matrix in the desired
    // pattern
    int limit = depth - 1;
    for (int i = 0; i < depth; i++)
    {
        for (int j = 0; j < depth; j++)
        {
            if (i + j < limit)
            {
                matrix[i][j] = 1;
            }
            else if (i + j > limit)
            {
                matrix[i][j] = -1;
            }
            else
            {
                matrix[i][j] = 0;
            }
        }
    }

}
```

Console

```
nidavellir :: UE20CS152/week5/pp1 » ./main
enter the limit >> 3
1 1 0
1 0 -1
0 -1 -1
nidavellir :: UE20CS152/week5/pp1 » ./main
enter the limit >> 5
1 1 1 1 0
1 1 1 0 -1
1 1 0 -1 -1
1 0 -1 -1 -1
0 -1 -1 -1 -1
nidavellir :: UE20CS152/week5/pp1 »
```

Practice Program 2

Write A Program to add two distances in the inch-feet system using structures.

`main.c`

```c
#include <stdio.h>
#include "utils.h"

int main()
{
    measure m1, m2;

    printf("The First Measure \n");
    load_measure(&m1);
    printf("The Second Measure \n");
    load_measure(&m2);

    // add them, accounting for overflow
    measure final = add(m1, m2);

    // display
    printf("The Final Distance is > ");
    display_measure(final);

    return 0;
}
```

`utils.h`

```c
struct measure
{
    float feet;
    float inches;
};

typedef struct measure measure;

void display_measure(measure number);
void load_measure(measure* number);
measure add(measure c1, measure c2);
```

`utils.c`

```c
#include <stdio.h>
#include "utils.h"


void load_measure(measure* number)
{
    // function to accept a measure from the user
    printf("Enter the Measure as '<feet> <inch>' >> ");
    scanf("%f %f", &number->feet, &number->inches);
}

void display_measure(measure number)
{
    printf("%f Feet and %f Inches.\n", number.feet, number.inches);
}

measure add(measure m1, measure m2)
{
    // adds two complex numbers
    measure m3 = {0, 0};
    m3.feet = m1.feet + m2.feet;

    float temp_inches = m1.inches + m2.inches;
    if (temp_inches >= 12)
    {
        m3.inches = temp_inches - 12.0;
        m3.feet += 1;
    }
    else
    {
        m3.inches = temp_inches;
    }
    return m3;
}
```

Console

```
nidavellir :: UE20CS152/week5/pp2 » ./main
The First Measure
Enter the Measure as '<feet> <inch>' >> 1 2
The Second Measure
Enter the Measure as '<feet> <inch>' >> 3 4
The Final Distance is > 4.000000 Feet and 6.000000 Inches.
nidavellir :: UE20CS152/week5/pp2 » ./main
The First Measure
Enter the Measure as '<feet> <inch>' >> 5 5
The Second Measure
Enter the Measure as '<feet> <inch>' >> 5 7
The Final Distance is > 11.000000 Feet and 0.000000 Inches.
nidavellir :: UE20CS152/week5/pp2 » ./main
The First Measure
Enter the Measure as '<feet> <inch>' >> 4 8
The Second Measure
Enter the Measure as '<feet> <inch>' >> 3 6
The Final Distance is > 8.000000 Feet and 2.000000 Inches.
nidavellir :: UE20CS152/week5/pp2 »
```