

Lab Assignment 5b

Akshat Mittal - 20107

June 2021

Contents

1. Linear Search in array
2. Rolling a dice 36000 times
3. Array Grades and its operations
4. Sum of two matrices
5. Transpose of a matrix
6. Product of two matrices
7. Matrix with element -1,0,1

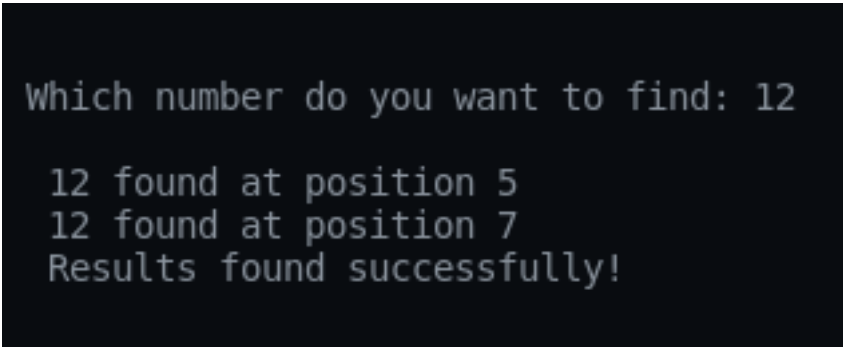
1

1.1 Code

```
#include <stdio.h>
int linear_search(int a[], int n)
{
    int i, k = 0;
    for (i = 0; i < 10; ++i)
    {
        if (a[i] == n) //compares the given number with every element of array
        {
            k = 1; //true if element found
            printf("\n %d found at position %d", n, i);
        }
    }
    return k;
}

int main()
{
    printf("\n\n\n");
    int n, a[10] = {2, 5, 17, 45, 23, 12, 78, 12, 16, 23};
    printf("Which number do you want to find: ");
    scanf("%d", &n); //input the number to be searched in the array
    if (linear_search(a, n)) //if function return true
        printf("\n Results found successfully!");
    else //if function return false
        printf("\n No results found for %d in array.", n);
    printf("\n\n\n");
    return 0;
}
```

1.2 Output

A screenshot of a terminal window with a dark background and light gray text. It shows the output of the program: "Which number do you want to find: 12", followed by "12 found at position 5", "12 found at position 7", and "Results found successfully!".

```
Which number do you want to find: 12

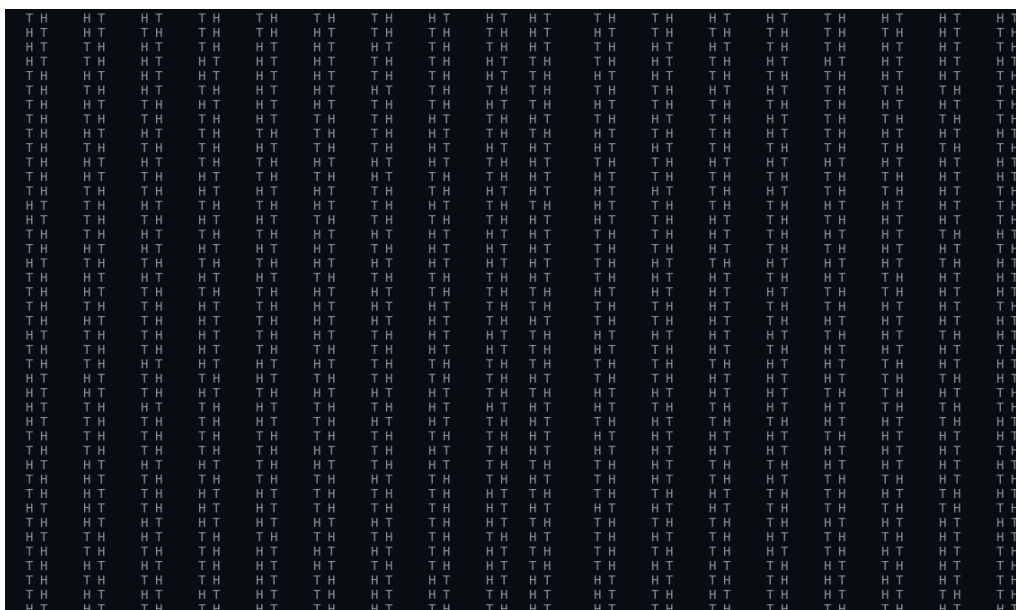
12 found at position 5
12 found at position 7
Results found successfully!
```

2

2.1 Code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("\n\n\n");
    int r, i;
    char a[2][36000];
    for (i = 0; i < 36000; i++)
    {
        r = (rand() % 2);    //assign 0 or 1 on random to r
        if (r)
        {
            a[0][i] = 'H';
            a[1][i] = 'T';
        }
        else
        {
            a[0][i] = 'T';
            a[1][i] = 'H';
        }
        printf("%c %c", a[0][i], a[1][i]);    //print the result of 1 toss of both dice
        printf("\t");
    }
    printf("\n\n\n");
    return 0;
}
```

2.2 Output

The output is a large grid of 36000 pairs of characters, each pair representing the result of a toss of two dice. The characters are 'H' and 'T', representing heads and tails. The grid is organized into 10 columns and 3600 rows. Each row contains 10 pairs of characters, separated by a tab character. The pairs are: (H, T), (T, H), (H, T), (T, H), (H, T), (T, H), (H, T), (T, H), (H, T), (T, H). This pattern repeats for all 3600 rows, resulting in a total of 36000 pairs of characters.

3

3.1 Code

```
#include <stdio.h>
#include <stdlib.h>
float mean(int a[], int n)
{
    int s = 0, i;
    for (i = 0; i < n; ++i)
        s += a[i];           //sum all the numbers in the array
    return (((float)s) / n); //return sum/n;
}
int main()
{
    printf("\n\n\n");
    int i, j;
    int Grades[5][20]; //3a
    printf("\na) Array is initialized as Grades[5][20]");

    printf("\nb) There are %d rows in the array.", 5); //3b

    printf("\nc) There are %d columns in the array.", 20); //3c

    printf("\nd) There are %d elements in the array.", 5 * 20); //3d

    printf("\ne) Names of all elements in the first column of array: ");
    for (i = 0; i < 5; ++i)
        printf("%dx0, ", i); //3e

    printf("\nf) Name of the element in the third row and second column of the array: %dx0", Grades[2][1]);

    Grades[0][1] = 100; //3g
    printf("\ng) Element in the first row and second column: %d", Grades[0][1]);

    double mathGrades[20];
    for (i = 0; i < 20; ++i)
        mathGrades[i] = (rand() % 9) + 1; //assign random grades to mathGrades from 1-9

    printf("\n\nh) Enter the elements of array Grades: \n");
    for (i = 0; i < 5; ++i)
    {
        printf("Row %d: \n", i + 1);
        for (j = 0; j < 20; ++j)
            scanf("%d", &Grades[i][j]); //3h
    }

    printf("\ni) The array after initialization with 0: \n");
    for (i = 0; i < 5; ++i)
    {
```

```

    for (j = 0; j < 20; ++j)
    {
        Grades[i][j] = 0; //3i
        printf("%d ", Grades[i][j]);
    }
    printf("\n");
}

printf("\nj) First row of Grades after copying from mathGrades: \n");
for (i = 0; i < 20; ++i)
{
    Grades[0][i] = mathGrades[i]; //3j
    printf("%d ", Grades[0][i]);
}

printf("\n\nk) The greatest number in the first row is: ");
int big = Grades[0][0];
for (i = 0; i < 20; ++i)
    if (Grades[0][i] > big)
        big = Grades[0][i];
printf("%d", big);

printf("\n\nl) Elements of column 2 of array: ");
for (i = 0; i < 5; ++i)
    printf("%d ", Grades[i][1]);

printf("\n\nm) The average of the elements in the first row: %f", mean(Grades[0], 20));

printf("\n\nn) The final array in tabular form: \n");
for (i = -1; i < 5; ++i)
{
    for (j = -1; j < 20; ++j)
    {
        if (i == -1)
        {
            if (j == -1)
                printf(" ");
            else
                printf(" %c ", j + 65);
        }
        else
        {
            if (j == -1)
                printf("%d ", i + 1);
            else
                printf(" %d ", Grades[i][j]);
        }
    }
    printf("\n");
}

```

```

printf("\n\n\n");
return 0;
}

```

3.2 Output

```

a) Array is initialized as Grades[5][20]
b) There are 5 rows in the array.
c) There are 20 columns in the array.
d) There are 100 elements in the array.
e) Names of all elements in the first column of array: 0x0, 1x0, 2x0, 3x0, 4x0,
f) Name of the element in the third row and second column of the array: 2x1
g) Element in the first row and second column: 100

```

h) Enter the elements of array Grades:

Row 1:

1 4 2 3 2 6 4 8 6 8 5 7 3 6 9 4 7 6 3 7

Row 2:

5 2 6 3 7 5 8 9 4 3 6 7 2 6 4 6 8 5 7 5

Row 3:

4 6 7 3 6 3 1 5 7 3 6 8 4 8 6 4 3 1 5 8

Row 4:

4 7 4 2 4 2 8 5 7 4 2 6 1 6 3 8 5 3 9 6

Row 5:

9 6 9 6 9 0 5 3 7 5 7 5 3 1 4 5 8 0 6 4

i) The array after initialization with 0:

```

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

```

j) First two of Grades after copying from mathGrades:

2 8 1 8 6 8 2 4 7 2 6 5 6 8 6 5 7 1 8 2

k) The greatest number in the first row is: 8

l) Elements of column 2 of array: 8 0 0 0 0

m) The average of the elements in the first row: 5.100000

n) The final array in tabular form:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1)	2	8	1	8	6	8	2	4	7	2	6	5	6	8	6	5	7	1	8	2
2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4

4.1 Code

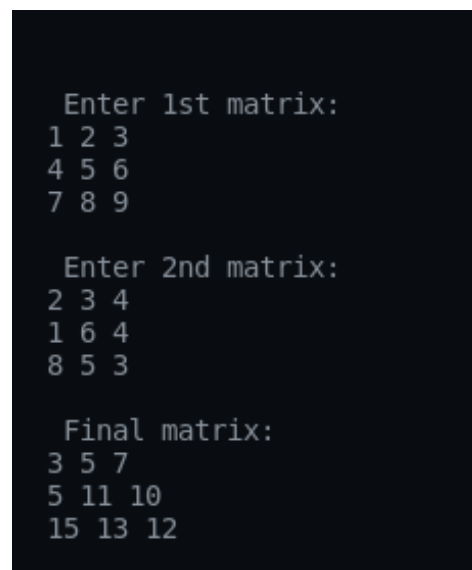
```
#include <stdio.h>
int main()
{
    printf("\n\n\n");
    int a[3][3], b[3][3], i, j;

    printf("\n Enter 1st matrix: \n");
    for (i = 0; i < 3; ++i)
        for (j = 0; j < 3; ++j)
            scanf("%d", &a[i][j]); //input 1st matrix element-wise

    printf("\n Enter 2nd matrix: \n");
    for (i = 0; i < 3; ++i)
        for (j = 0; j < 3; ++j)
            scanf("%d", &b[i][j]); //input 2nd matrix element-wise

    printf("\n Final matrix: \n");
    for (i = 0; i < 3; ++i)
    {
        for (j = 0; j < 3; ++j)
            printf("%d ", a[i][j] + b[i][j]); //print the addition of both matrices
        printf("\n");
    }
    printf("\n\n\n");
    return 0;
}
```

4.2 Output



The screenshot shows the output of the C program. It displays three 3x3 matrices. The first matrix is entered as 1 2 3, 4 5 6, 7 8 9. The second matrix is entered as 2 3 4, 1 6 4, 8 5 3. The final output matrix is the sum of the first two, showing 3 5 7, 5 11 10, and 15 13 12.

```
Enter 1st matrix:
1 2 3
4 5 6
7 8 9

Enter 2nd matrix:
2 3 4
1 6 4
8 5 3

Final matrix:
3 5 7
5 11 10
15 13 12
```

5

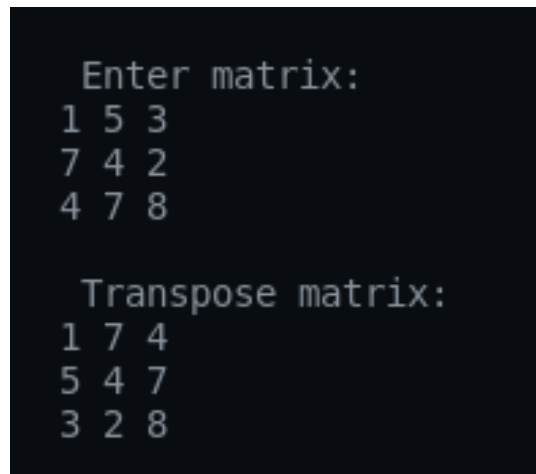
5.1 Code

```
#include <stdio.h>
int main()
{
    printf("\n\n\n");
    int a[3][3], i, j;

    printf("\n Enter matrix: \n");
    for (i = 0; i < 3; ++i)
        for (j = 0; j < 3; ++j)
            scanf("%d", &a[i][j]); //input matrix element-wise

    printf("\n Transpose matrix: \n");
    for (i = 0; i < 3; ++i)
    {
        for (j = 0; j < 3; ++j)
            printf("%d ", a[j][i]); //print by transposing every element
        printf("\n");
    }
    printf("\n\n\n");
    return 0;
}
```

5.2 Output



The screenshot shows the output of the C program. It first prompts 'Enter matrix:' and then displays a 3x3 matrix of integers: 1 5 3, 7 4 2, and 4 7 8. After a blank line, it prompts 'Transpose matrix:' and displays the transposed 3x3 matrix: 1 7 4, 5 4 7, and 3 2 8.

```
Enter matrix:
1 5 3
7 4 2
4 7 8

Transpose matrix:
1 7 4
5 4 7
3 2 8
```


6

6.1 Code

```
#include <stdio.h>
int main()
{
    printf("\n\n\n");
    int a[10][10], b[10][10], c[10][10], i, j, k, m, n, o, p;

    printf("\n Enter the order of 1st matrix: ");
    scanf("%d%d", &m, &n);
    printf("\n Enter 1st matrix: \n");
    for (i = 0; i < m; ++i)
        for (j = 0; j < n; ++j)
            scanf("%d", &a[i][j]); //input 1st matrix element-wise

    printf("\n Enter the order of 2nd matrix: ");
    scanf("%d%d", &o, &p);
    printf("\n Enter 2nd matrix: \n");
    for (i = 0; i < o; ++i)
        for (j = 0; j < p; ++j)
            scanf("%d", &b[i][j]); //input 2nd matrix element-wise

    if (n != o)
        printf("\n Matrix multiplication not possible."); //if no. of columns of 1st matrix is not equal to no. of rows of 2nd matrix
    else
    {
        printf("\n Final matrix: \n");
        for (i = 0; i < m; ++i)
        {
            for (j = 0; j < p; ++j)
            {
                c[i][j] = 0;
                for (k = 0; k < n; ++k)
                    c[i][j] += a[i][k] * b[k][j]; //multiplying row of first matrix to column of second matrix
                printf("%d ", c[i][j]);
            }
            printf("\n");
        }
    }
    printf("\n\n\n");
    return 0;
}
```

6.2 Output

```
Enter the order of 1st matrix: 2 3

Enter 1st matrix:
1 4 2
2 5 1

Enter the order of 2nd matrix: 3 2

Enter 2nd matrix:
2 5
3 5
2 1

Final matrix:
18 27
21 36
```

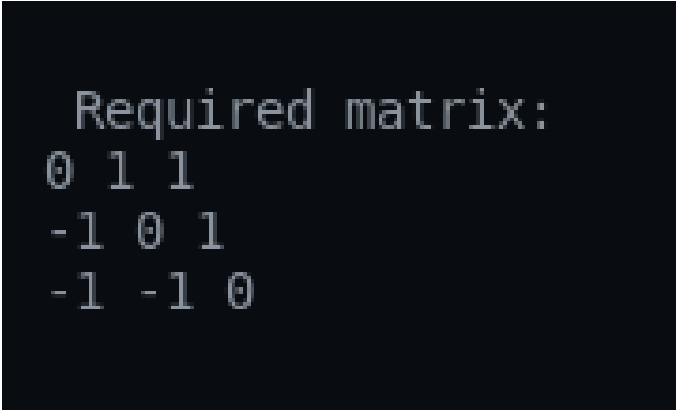
7

7.1 Code

```
#include <stdio.h>
int main()
{
    printf("\n\n\n");
    int a[3][3], i, j;
    printf("\n Required matrix: \n");
    for (i = 0; i < 3; ++i)
    {
        for (j = 0; j < 3; ++j)
        {
            if(i>j)a[i][j]=-1; //for lower diagnol triangle
            else if(i<j)a[i][j]=1; //for upper diagnol triangle
            else a[i][j]=0; //for principle diagnol

            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
    printf("\n\n\n");
    return 0;
}
```

7.2 Output



```
Required matrix:
0 1 1
-1 0 1
-1 -1 0
```