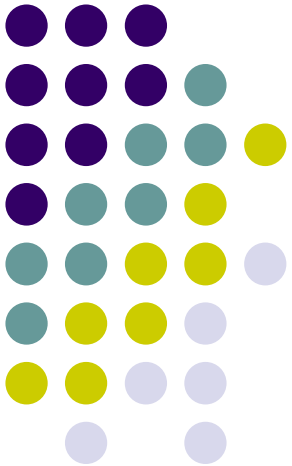
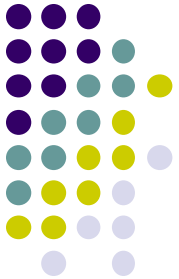
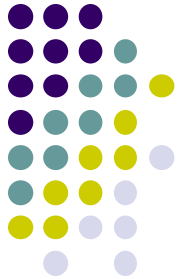
PDS Lab Section 15

February 4, 2021





General Instructions

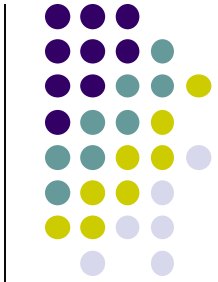


- Add header in front of your program

```
/*
*     Section 15
*     Roll No  :  20CS30010
*     Name     :  Your Name
*     Assignment No : 3
*     Description : Program to check points
*/
```

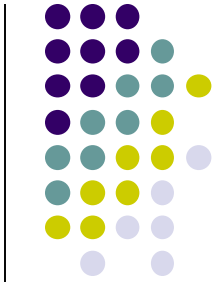
- Name your file with assgnX_Y.c, where X is the assignment number and Y is your roll no.
  - assgn9_20ME10010, assgn10_20CE30014,…..

Indenting your program



- Give proper indentation (space in front of a line) in your program to make your program look nice
- Indent every block in your program consistently
  - What is a block? – statements inside if, else, for, while etc…(for now)
  - Idea is to be able to see which statements are part of which if or which else or which for/while etc. easily
    - Helps in making less mistakes in missing braces etc. when you write programs
    - Helps in easier debugging if your program does not work

Badly indented program

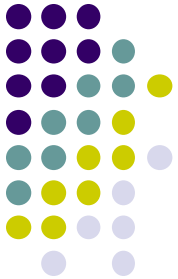


```
int main()
{
int i, j,k,n =10,sum, count;
for (i=0; i<n; i++)
{
sum=0; count=0;
for (j=0;j<n;j++)
{ scanf("%d",&k);
if (k>0)
{sum = sum + k;
 count = count + 1;
}
printf("Sum=%d Count=%d\n", sum, count);
}
}
}
```

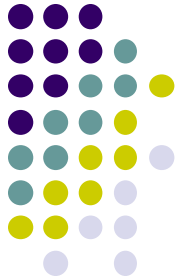
Properly Indented Program

```
int main()
{
    int i, j,k,n = 10, sum, count;
    for (i=0; i<n; i++)
    {
        sum=0;
        count = 0;
        for (j=0; j<n; j++)
        {
            scanf("%d",&k);
            if (k>0)
            {
                sum = sum + k;
                count = count + 1;
            }
            printf("Sum=%d Count=%d\n", sum, count);
        }
    }
}
```

Commenting a function



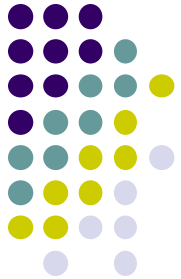
- Before each function declaration, add a comment with the following
  - The name of the function
  - A one line description for each parameter
  - A one line description of the return value
  - A small description of what the function does



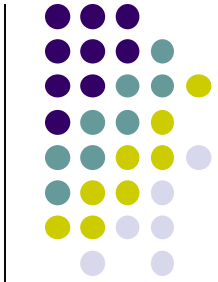
- Example:

```
/*****************
Function name: FindMin
Parameters:
        A : array of integers
        n : no. of integers in the array A
Return value: The minimum integer in array A
Description: This function finds the minimum value of a set of
             integers stored in array A
*****************/

int FindMin(int A[ ], int n)
{
   ….
}
```

Intermediate Submission for the First Assignment

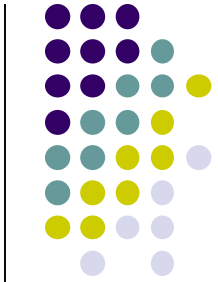


- For the first assignment, you will see two submission links in moodle
  - Assignment 13 Submission (Intermediate)
  - Assignment 13 Submission (Final)
- You **must submit** the .c file containing whatever you have done so far for Assignment 11 (first assignment today) in the link "Assignment 13 Submission (Intermediate)" strictly between 10-30 am to 10-45 am (**the link will open at 10-30 am and close at 10-45 am**)
  - Submit whatever you have done till then, we want to see how regularly you are progressing
  - **Not submitting will incur 30% penalty irrespective of what your final submitted version is**
- Submit the final .c file using the link "Assignment 13 Submission (Final)" as usual anytime before the lab ends (**must submit for grading, only the final version will be graded**)



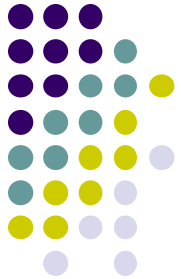
Today’s New Topic: 2-d Array

2-d Arrays



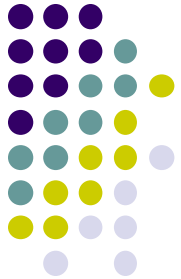
```
int main()
{
   int A[10][10];  /* declares a 10x10 array */
   int i, j;

   /* read each element of the array from keyboard */
   for (i = 0; i < 10; i++)
       for (j = 0; j < 10; j++)
             scanf("%d", &A[i][j]);

   /* find the largest element in the array */
   max = A[0][0];
    for (i = 0; i < 10; i++)
       for (j = 0; j < 10; j++)
             if (max < A[i][j]) max = A[i][j];

   printf("Largest no. in the 2-d array is %d\n", max);
}
```

Things To Remember



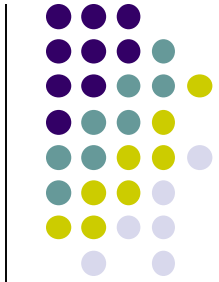
- While declaring, need to specify both dimensions
  - int A[100][100], B[20][20],…
- The two dimensions need not be equal
  - int X[10][20], B[25][25]
- int A[10][20] is actually a block of 10x20 = 200 integers
  - A[0][0], A[0][1], A[0][2],…,A[0][19]
  - A[1][0], A[1][1], A[1][2], …,A[1][19]
  - A[2][0], A[2][1], A[2][2], …,A[2][19]
  - …
  - A[9][0], A[9][1], A[9][2], …,A[9][19]

Printing 2-d Arrays



```
int main()
{
   int A[10][10];  /* declares a 10x10 array */
   int i, j, n, m;

   /* read the no. of rows and columns in the matrix */
   printf("Enter the no. of rows and columns: ");
   scanf("%d%d", &n, &m);

   /* read each element of the array from keyboard */
   for (i = 0; i < n; i++)
       for (j = 0; j < m; j++)
             scanf("%d", &A[i][j]);

   /* print the array nicely*/
    for (i = 0; i < n; i++)
   {
       for (j = 0; j < m; j++)
          printf("%6d", A[i][j]);
       printf("\n");
   }
}
```

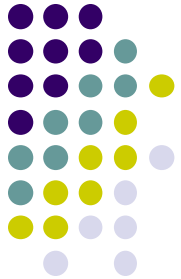
Passing a 2-d array to a function



- Similar to passing a 1-d array, but you must specify the second dimension
- Calling the function is same as for 1-d array

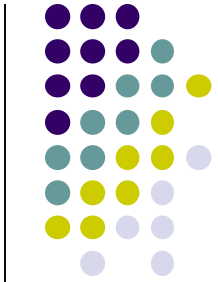
```
/* Print the elements of a m x n array */
void PrintArray(int A[ ][10], int m, int n)
{
  int i, j;
  for (i = 0; i < m; i++)
  {
    for (j = 0; j < n; j++)
       printf("%6d", A[i][j]);
    printf("\n");
  }
}
```

Printing 2-d Arrays



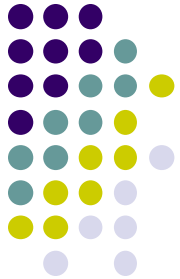
```
int main()
{
   int A[10][10];  /* declares a 10x10 array */
   int i, j, p, q;

   /* read the no. of rows and columns in the matrix */
   printf("Enter the no. of rows and columns: ");
   scanf("%d%d", &p, &q);

   /* read each element of the array from keyboard */
   for (i = 0; i < p; i++)
       for (j = 0; j < q; j++)
             scanf("%d", &A[i][j]);

   /* print the array nicely*/
   PrintArray(A, p, q);
}
```

Practice Problem 1



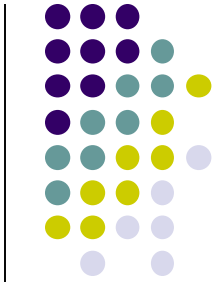
- A n x m matrix can be represented by a n x m 2-d array.
- Write a C program that does the following:
  - Declare a 2-d array X of size 20 x 30
  - Reads in two integers, n and m (0 < n < 20 and 0 < m < 30)
  - Read in the elements of a n x m matrix in the 2-d array X
  - Print the matrix X in a matrix form
  - Find the number of elements k in the matrix that are divisible by 3
  - Print k

Practice Problem 2



- Write a C program that does the following:
  - Declare two 2-d arrays X and Y of size 10 x 10
  - Reads in an integer n (0 < n < 10)
  - Read in two n x n matrices in the 2-d arrays X and Y
  - Print the matrices X and Y in matrix form
  - Add the two matrices and store the sum in X (i.e., X = X + Y)
  - Print the matrix X after summing

Practice Problem 3



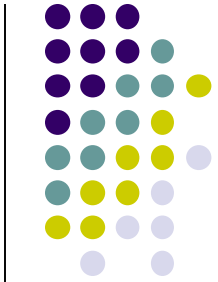
- Write a C function

  void transpose(int A[ ][10], int m, int n)

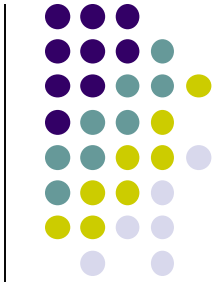
that takes a m x n matrix as parameter and converts it to its transpose

- Write a main function that reads in a m x n matrix (m, n < 10) and calls the transpose() function to compute its transpose, and then print it

Practice Problem 4

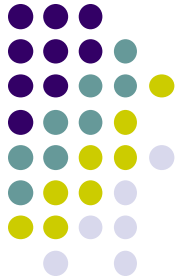


- An upper triangular matrix is a square matrix for which all elements below the diagonal are 0
- Write a C function checkUpper() with appropriate parameters and return values that takes as parameter an n x n matrix and returns 1 if it is upper triangular, 0 otherwise
- Write a main function that reads in a n x n matrix (n < 10) and calls the checkUpper() function to find print whether the matrix is upper triangular or not

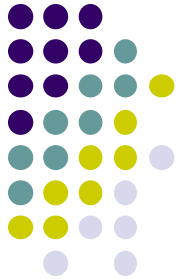


Assignments

Assignment 15



- XYZ Pvt. Ltd. stores the following information for each of its employees:
  - Name (max 20 characters including ‘\0’ at end), employee code (integer), age (integer), salary (floating point)
- Define a C structure named employee to store this data for each employee

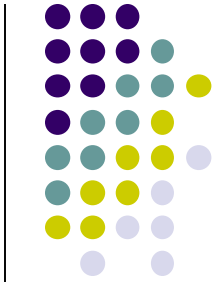


- Write a function

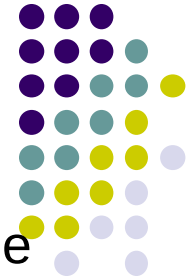
  int SearchEmpl(struct employee A[ ], int n, …)

that will search for an employee with a specific employee code, and if found, will return his/her name, age and salary.

- A is the array of employee structures containing the records of all employees, and n is the number of such structures in A
- The return value should be 1 if the employee code matches a structure, 0 otherwise
- You will need to add other parameters required to return the age, salary and name

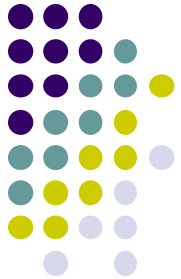


- Write a main function that does the following:
  - Declares a pointer p to point to an employee structure
  - Reads in the number of employees n from the keyboard
  - Mallocs an array of n number of employee structures and stores the return value of malloc in p
    - i.e., holds the information for n employees, one employee structure for each employee
    - This is your array of structures
  - Reads in the name, employee code, age, and salary for the n employees one by one in a loop
    - When you read the information for one employee, call the SearchEmpl() function to check if the employee code already exists. If yes, reject the information entered with a suitable message and ask the user to enter the information for this employee again.
    - So finally you should have entered the information of n employees with distinct employee codes



- Reads in a floating point number k and print the name, employee code, age, and salary of all employees with salary > k
  - Compare the salary field of the employee structure of each employee with k
  - Print the name, employee code, age, and salary fields if salary field is > k
  - Print the information for each such employee in a separate line
  - If there is no employee with salary > k, print a suitable message saying so
- Reads in an integer x
- Call the SearchEmpl() function with appropriate parameters to find if there is any employee with employee code x. If yes, print his/her name, age, and salary with an appropriate message
  - If there is no employee with employee code x, print a suitable message saying so

Example



- n = 4,
- employee details read in array (<employee code, name, age, salary>)

  <1002, David, 32, 30000>

  <1010, John, 36, 40000>

  <1001, Mary, 35, 42000>

  <1005, Tom, 40, 50000>
- k = 40000
- x = 1001

Your program should print

Employees with salary greater than 40000 are

1001, Mary, 35, 42000

1005, Tom, 40, 50000

Employee with employee code 1001 is Mary, 35, 42000

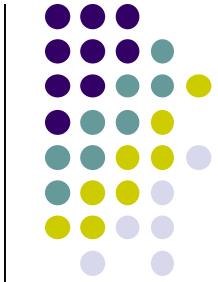
Assignment 16

- Write a function

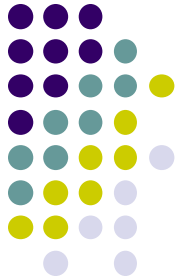
  void rotate(int A[ ][10], int m, int n, int type)

that takes a m x n 2-d array A, and does the following

- If type = 1, cyclically rotates each of its rows to the right
  - If the elements in a row are a1, a2, a3, a4, then after the above rotation, they will become a4, a1, a2, a3
  - This needs to be done for each row
- If type = 2, cyclically rotates each of its columns down
  - If the elements in a column are $\begin{matrix} a1 \\ a2 \\ a3 \\ a4 \end{matrix}$ then after the above rotation, they will become $\begin{matrix} a4 \\ a1 \\ a2 \\ a3 \end{matrix}$
  - This needs to be done for each column
- You can assume that value of type will be 1 or 2 only



- Write a main function that:
  - Reads in the dimensions m and n of the 2-d array (m, n < 20)
  - Reads in all array elements row by row (so elements of 1st row are all read before any element of 2nd row is read and so on)
  - Prints the 2-d array nicely
  - In a while loop (or do-while loop), does the following:
    - Read in an integer opt
    - If opt = 0, exit the program
    - If opt = 1, cyclically right rotate all rows of the 2-d array, and then print the new array
    - If opt = 2, cyclically rotate down all columns of the 2-d array and then print the new array
    - Call the rotate() function with appropriate parameters to rotate the elements
    - Note that each time the original array is changed. And the next rotation, if any, is done on the new array. See the example in the next slides



- Suggestion to start
  - Write the code (both main() and the rotate() function) first assuming you have only two options for opt, 0 and 1 only
  - Test everything to make this work
  - Then add the code to handle opt = 2
  - You may also want to write first a simple function that takes the 2-d array and its dimensions as parameters and prints it nicely, so that you do not have to write that code again and again in main()

Example

- m = 3, n = 3
- Initial array read

2	1	3
9	5	7
3	6	8

- Entered opt = 1
- New array after right rotate of rows

3	2	1
7	9	5
8	3	6

- Entered opt = 1
- New array after right rotate of rows

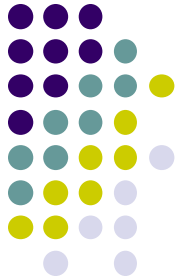
1	3	2
5	7	9
6	8	3

- Entered opt = 2
- New array after down rotate of columns

6	8	3
1	3	2
5	7	9

- Entered opt = 0
  - Exit the program

Teaser problem (not to be submitted)



- Write a program to read in an integer n (n < 20) and given an n by n chess board, find the minimum number of chess Queens so that all the board places are 'attacked' by the Queens. Print the output of the Queens placed using an appropriate print statement.