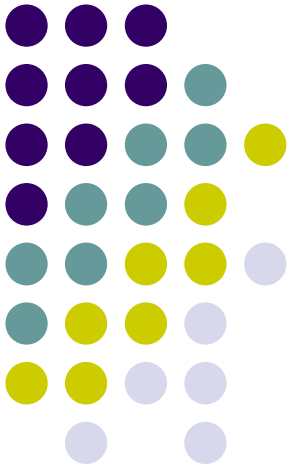
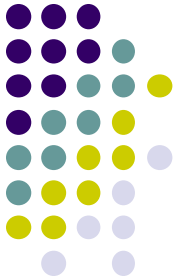
PDS Lab Section 15

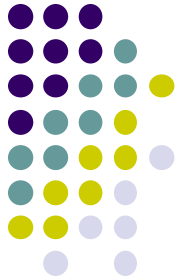
January 7, 2021



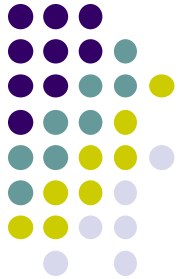


Instructions for Lab Test

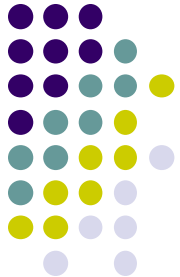
Lab Test 1



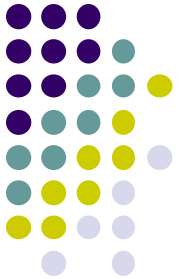
- Date: January 14th, 2021
- Syllabus: Upto Arrays (no functions)
- Make sure you have a good network connection throughout the test
  - TAs may ask you to show your programs any time, you must be able to show it if they ask
- Join in Teams by 9 am in General channel as usual. After we tell you to, go join in your respective TA channels immediately. Test starts strictly at 9-10 am
- **TAs will not answer ANY questions at all on that day, so do not ask them anything.** You will work completely on your own. If you have any questions on understanding the problems, send a private chat message to the teachers. Also, watch your channel for any announcement
- Read the Malpractice Guidelines document sent to you earlier carefully. Any case of malpractice caught will incur an immediate **0 for the whole test** and possible additional penalty.



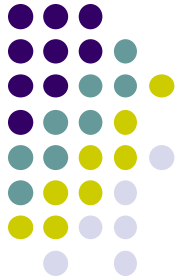
- Format: There will be two parts with two problems
- Each Part:
  - Time is 1 hour 10 minutes
    - Do NOT think of it as a 1 hour 10 minute exam. Think of it as 5 minute to read the assignment carefully, 1 hour to do it, 5 minute to submit
  - Start submitting early, no excuses for not submitting in moodle. If you wait till the last minute and you have a network problem, nothing can be done. Email submission will incur heavy penalties, no excuses at all.
  - Similar to class, you will have to do an intermediate submission for EACH part. See next slide for timings. This is a must, not submitting will incur 30% penalty



- Timings
  - Part 1:
    - Problem statement available in moodle: 9-10 am
    - Intermediate submission (through intermediate submission link): between 9-45 am and 9-55 am
    - Final Submission (through final submission link): by 10-20 am (strict, no extensions will be given)
  - 10 minute break
  - Part 2:
    - Problem statement available in moodle: 10-30 am
    - Intermediate submission (through intermediate submission link): between 11-00 am and 11-10 am
    - Final Submission (through final submission link): by 11-40 am (strict, no extensions will be given)



General Instructions

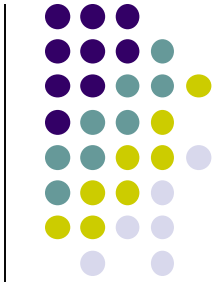


- Add header in front of your program

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

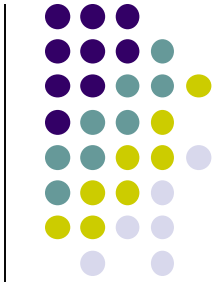
- Name your file with assgnX_Y.c, where X is the assignment number and Y is your roll no.
  - assgn9_20ME10010, assgn10_20CE30014,…..

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

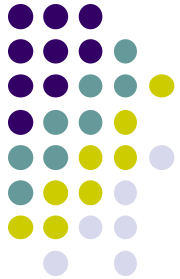


```
int main()
{
int i, j,k,n =10,sum, count;
for (i=0; i<n; i++)
{
sum=0; count=0;
for (j=0;j<n;j++)
{ scanf("%d",&k);
if (k>0)
{sum = sum + k;
 count = count + 1;
}
printf("Sum=%d Count=%d\n", sum, count);
}
}
}
```

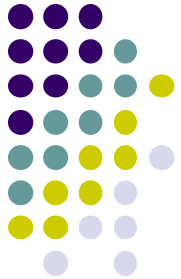
Properly Indented Program

```
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);
        }
    }
}
```

Intermediate Submission for the First Assignment

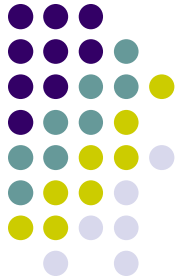


- For the first assignment, you will see two submission links in moodle
  - Assignment 9 Submission (Intermediate)
  - Assignment 9 Submission (Final)
- You **must submit** the .c file containing whatever you have done so far for Assignment 9 (first assignment today) in the link "Assignment 9 Submission (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 9 Submission (Final)" as usual anytime before the lab ends (**must submit for grading, only the final version will be graded**)



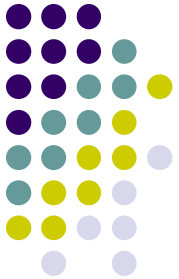
Today’s Topic: Functions

Functions



- Define something once, call it again and again whenever needed
- Defining/Declaring the function
  - The actual body of the function
  - C program to actually do what the function is supposed to do
- Calling a function
  - Call the function to do what you want it to do
  - Can be called at different times to do the same thing on different data

Function Declaration



```
int  IsPrime(int n)
{
      int i = 2, flag = 1;
      if (n == 1) return 0;
      while (i < n/2 && flag == 1)
      {
          if (n % i == 0) flag = 0;
          i = i + 1;
      }
      return (flag);
}
```

int : type of the return value (says that the function will return one integer value, in this case 1 if the number is prime, 0 otherwise)

IsPrime : Name of the function; we will call it by this name

int n : formal parameter (says that this function will take one integer from outside and do something with it, in this case, check if it is prime or not)

Function Call

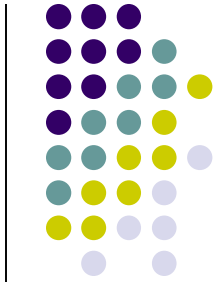
```
int main()
{
    int i, n, flag;
    scanf("%d", &n);
    for (i=2; i <= n; i++)
    {
        flag = IsPrime(i);   /* i is the actual parameter passed */
        if (flag == 1) printf("Prime no. = %d\n", n);
    }
}
```

This will print all prime numbers between 2 and n

Note:

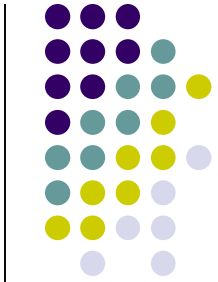
- Formal parameter and actual parameter names may be same or different
- The same variable name can be used in main and in other functions (for example, we used i in both)
  - But the two i's are two completely different variables

Another Example: Larger of 2 No.s



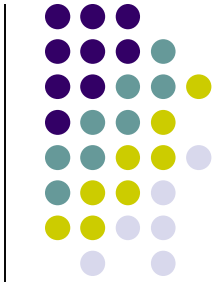
```
int main()
{
        int a, b, large;
        scanf("%d%d", &a, &b);
        large = LargerOf(a, b);
        printf("The larger of %d and %d is %d\n", a, b, large)
}

int LargerOf(int x, int y)
{
        if (x > y ) return x;
        else return y;
}
```

Passing Arrays as Parameters to Functions

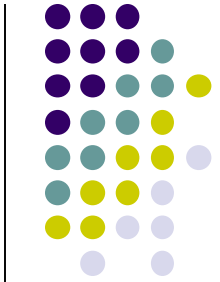


```
int FindMin(int X[ ], int n);    /* prototype declaration */

int main()
{
    int i, A[100], min;
    scanf("%d", &n);
    for (i=0; i<n; i++)
        scanf("%d", &A[i]);
    min = FindMin(A, n);
    printf("The minimum value is %d\n", min);
}

Int FindMin(int X[ ], int n)
{
    int i, min=X[0];
    for (i=1; i<n; i++)
        if ( X[i] < min) min = X[i];
    return(min);
}
```

Things To Remember

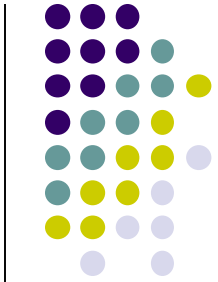


- In the declaration:
  - The dimension of the array need not be specified in the declaration, but [ ] must be there
    - int A[ ], float B[ ] …
  - If you pass an array as a parameter, there must be another int parameter to pass the actual number of elements in the array, so always make sure to add it
- The call just specifies the name of the array (no [ ])
  - FindMin(A, n)
- Changes to the array made inside the function are made to the original array passed as parameter
  - Unlike other variables, where change is made to a copy of the original variable, and the original variable remains unchanged

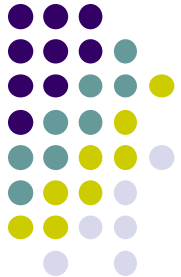


Assignments

Assignment 9



- Write a C function named int OddDigitCount(int n)
  - The function takes as parameter one integer n, and returns the number of odd digits in n
    - Note that for any integer k, k%10 gives the last digit, and k/10 gives the number without the last digit
      - Example, if k = 746, then k%10 = 6 (= last digit), and k/10 = 74 (= 746 without the last digit) in C (and not 74.6 as int/int division truncates in C). You can now do 74%10 = 4 to get the next digit and so on. Extract the digits in a while loop and check them, keeping count of the odd digits found. Work out by hand with an actual number first to check till how far the loop should go.
  - For example, if the parameter passed (n) is 1323, the function returns 3 as there are 3 odd digits, 1, 3, and 3 in 1323. Similarly, if the parameter passed (n) is 46, the function should return 0 as there are no odd digits in 46



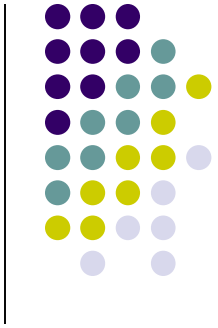
- Write a main function that
  - Reads in an integer n (assume n < 100)
  - Reads in n integers in an array A (in a loop)
  - Prints the n integers in the array A (in a separate loop from read)
  - Prints the number of odd digits in each number in A and also, the number in A with the maximum number of odd digits
    - Use another loop that calls the function OddDigitCount() with each element of the array as parameter, and prints the return value of the function for each element
    - Remember the maximum count found while looping, print it after coming out of the loop
- For example, if you read in 34, 1323, 50, 71 in array A, your output should look like (number of spaces can vary):

34 1323 50 71

1 3 1 2

The number with maximum no. of odd digits is 1323

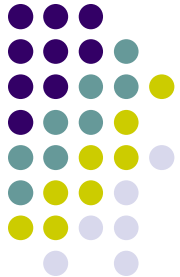
Assignment 10



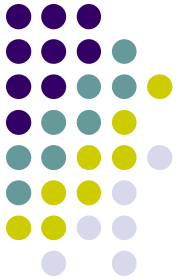
- Write C function

  int CheckStr(char S1[ ], char S2[ ])

  that takes as parameter two strings S1 and S2 of characters (null-terminated, i.e., terminated by '\0'), and returns 1 if there exists any combination of two characters in S1 (not necessarily consecutive) such that the sum of their ascii codes is the same as the maximum ascii code of any character in S2.
- Example:
  - Suppose S1 = "023AZC" and S2 = "7pq". Then maximum ascii code of any character in S2 is 113 (the ascii code of 'q'). Then the function should return 1 as there exists a combination of two characters in S1, '0' (ascii code 48) and 'A' (ascii code 65) such that their sum is = 48 + 65 = 113
  - Suppose now S1 is "02BZC". S2 is the same. Then the function should return 0 as there is no such combination of two characters in S1 whose ascii code sum is = 113

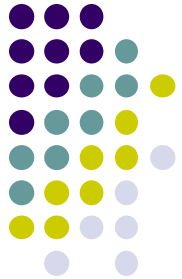


- To write CheckStr(), do the following in this order:
  - Find the lengths of S1 and S2 and store in two variables. Print the lengths of S1 and S2 with a message
  - Find the character with the maximum ascii code value in S2. Print both the character and its ascii code with a message
  - Look at all possible pairs of characters in S1 (two nested loops). For each pair considered inside the loop:
    - Sum the ascii codes of the two characters, and print the two characters and their ascii code sum
    - Then compare the sum with the max ascii code value of S2 found earlier
      - If found, return 1 immediately
      - If not found, go to the next pair and check
    - Return 0 at end of the function after coming out of al loops if no pair is found



- Write a main function that
  - Reads two strings X1 and X2 from the keyboard using %s in scanf. Assume that the strings will be of less than 100 characters each
  - Calls the function CheckStr() with X1 and X2 as parameters, and prints a nice message based on the return value
    - It is ok to print just whether there exists such a combination, the actual combination need not be printed if the function returns 1
- Do NOT use the ascii code values directly (like 48, 49, 65 etc.) anywhere in your program, 50% penalty if used
  - Your don't need to, as characters are stored inside by their integer ascii codes, and can be compared or added directly (like 'a' + 'c', if (ch > 'a') etc.

Teaser problem (not to be submitted)



- Read in two character strings s and p. Read in integers n and k. Find all matches between s and p of length at least n which have at most k mismatches. You can use functions but no additional array other than s and p.