Experiment no. → 1

* AIM

WAP to print this is AI - DS B2 in C.

* Software Used

Code Blocks (GNU GCC min GW compiler.)

* Theory

C is a strongly typed case sensitive language and follows a very defined structure. Each C program has a few mandatory Component - A c program starts with a pre processor directive # and includes one header file which is stdioh; this header file contains various input output functions req. for the execution of a program. Each C program should have one function which is named "main". The execution of a program starts from the first line of main and ends with the last line of main. Blocks in C language are contained inside parenthesis { }. Each Statement in C language end with a semi colon ;

* Code / Program

```
# include <stdio.h>
void main ()
{
Print f ("this is AI - DS B2");
getch ();
}
```

[illegible]

Teacher's Signature

Experiment no. 2

* AIM

write a program to print ASCII value of a charachter.

* Software Used

code block (GNU GCC MINGW COMPILER)

* Theory

ASCII (American Standard code for interfernation interchange) is the most common character encoding format for next data is computer and on the internet. in Standard ASCII - enclosed data, there are unique value for 128 alphabetic, numeric or special additional character and control codes.

* Code / program

```
# include <stdio.h>
void main ( )
{
   char a ;
   print f ("enter character");
   scan f ("d.c" q & a);
   print f ("ascii value of character is xd", a);
   getch ( );
}
```

Date 5/Dec/22

Expt. No. Page No. 3

Experiment → 2

* AIM
write a Program to print sum and average of 3 integers input by the users.

* Software used
code block (GNU GCC MINGW COMPILER)

* Theory
- Scanf :- The scan f () function reads data form the standard input stream stdio into the location given. by each entry in arrgument, list, each arrgument must be a painter to a varriable with a type that corresponds to a type specifier in format string.
- Print f :- The print f () function sends a formatted string to the standard output the (display). This string can display formatted varriable and special central characters.

* Code / Program

```
## include <stdio.h>
 int main ()   {
 float . a, b, c, sum , avg;
 prints ("enter 3 integers = ");
 scan f ("%f %f %f, $a ,$b, $c);
 sum = a + b + c;
 avg = sum/3;
 print f (" The sum is % f and avg. is % f, sum, avg.);
   }
```

Teacher's Signature

Date 5/Dec/22

Expt. No. ______ Page No. 4

Experiment → 4

* AIM

WAP to print the sum of the individual digit of a three digit no. input by the user.

* Software Used

Code block (

* Theory

An arithemetic operator performs mathematical operation Such as addition, subtraction, multiplication, division etc on numeric values. (Constant and variables)

operators	meaning of operators
+	→ addition
−	→ subtraction
*	→ multiplication
/	→ Division
%	→ remainder after division

* Code / program

```
# Include <stadio.h>
void main () {
Int. a, o, t, h;
Print f ("enter number:");
scan f ("% d", o$ a);
 o = a % 10;
 t = (a/10) % 10;
  h = (a/100) % 10;
 print f ("sum of numbers; %d", o+t+h);
  }
```

Teacher's Signature [illegible]

Date 12/Dec/22

Expt. No.

Page No. 5

Experiment No. → 5

* AIM:-

WAP to ~~print~~ check whether entered number is prime or not.

* Software Used:-

Code Blocks (GNU GCC min GW compiler).

* Theory:-

In C programming, a loop is used to repeat a block of code until the specific condition is met for loop is a repetition control structure that allows you to efficiently ~~number~~ write a loop that needs to excute a specific number of times.

* Code:-

```
# include <stdio.h>
  void main () {

  int num, x, flag = 0;
  print f ("enter a number:");
  Scan f ("%d", & num);
   for (x = 2 ; x < num ; x ++){

   if (num % x = = 0) {

        flag = 1;

        break;
   }
  }
```

Teacher's Signature

Date

Expt. No.

```
if (flag==1) {
    print ("not prime");
  }
  else
  {
  printf ("prime");
  }
}.
```

[illegible]

Teacher's Signature

Date

Expt. No. Page No. 7

Experiment No. → 6

* Aim :-
WAP to find out whether an input Number is angstrom No. or Not.

* Software Used :-
Code block (GNU GCC prime No compiler).

* Theory :-
An armstrong number is one whose sum of digit raised to the power three equals the number itself. 371, for example is an armstrong number because 3**3 + 7**3 + 1**3 = 371.

* Code / Program :-

```
  # include < stdio.h >
   # include < math.h >
     void main ( )
 { int  num , sum , a , cound = 0 , 6 ;
        Print f ("Enter No. : ");
        Scan f (" %d "; &num);
             a = num;
             b = num;
          while ( b > 0 ) {
                 b / = 10
               Count ++ ; }
          while (Num > 0) {
               int t = num % 10 ;
               Sum t = Pow (t, count);
               num / = 10;
          }
```

Teacher's Signature

Date

Expt. No.

Page No.

```
if (sum = = 0) {
    printf (" Number is Armstrong ); }
else {
    Printf ("Number is not Armstrong"); }
    }
```

Teacher's Signature

Date

Expt. No.   Page No. 8

Experiment No. → 7

* Aim :-
write a program to create a calculater using switch case.

* Software used :-
Code Block (GNU GCC prime no. MiNGW Compiler).

* Theory :-
Switch case statement evaluates a given expression and based on the evaluated value (Matching a certain condition), it excecutes the statement associates with it. Basically, it is used to perform different action based on different condition (cases).

* Code/ Program :-

```
# Include <stdio.h>
    void main ()
  {
      char u = 'y', ch;
      while (u = ='y') {
      int num 1, num 2;
      Print f ("Enter 2 Number:");
      Scan f ("%d, %d", &num 1, &num 2);
      print f ("+) ADD\n-) Subtract\n*) multiplication\n/) division:");
      f flush (stdin);
      scan f ("% c" & ch);
      switch (ch) {
          case '+':
          print f ("%d", num 1+ num2);
          break;
```

Teacher's Signature

Date

Expt. No.

```
        case '-'
        printf ("%d", num1, num2);
        break;
  case '*'.
        printf ("%d", num1, num2);
        break;
  case '/':
     printf ("%f", (float) num1/((float) num2);
     break
    defalt: Printf ("No choice entered \n");
    }
     printf ("\n Do you want to continue? Y or N:");
     fflush (stdin);
     Scanf ("%c"; &u);
    }
    }
```

Teacher's Signature

Experiment No. → 8

* Aim:-
write a recursirce program to print factorial of a number.

* Software used :-
Code Block (GNU GCC MINGW compiler)

* Theory:-
Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

* Code / Program :-

```
# include <stdio.h>
 int fact (int n)
 {
   if (n = = 1) { return 1 ; )
    return n* fact (n-1);
  }
   void main ()
  {
    Int n;
    Printf ("enter number;");
    scanf ("%d" &n);
    printf (" factorial: ÷d", fact (n);

   }
```

[illegible]

Experiment No. → 9

* Aim :-
write a program to multiply two mentions and print their product.

* Software Used :-
code Blocks (GNU GCC Compiler MingGW)

* Theory :-
Multidiminsional Orrays:
if we want to store data. as a tabular form, like a table with rows and coloumns then multidiminsional orrays are used. It is basically on orrays of orrays. Arrays can have any number of dimension. The most common of them are 2D and 3D orrys. The syntax for difining 2-D arrys.

```
int mach( ) {
      int orr [no. of ~~coloumn~~ rows] [no. of coloumn] = {{ element },
                                         { element }, { elements }
   }
```

* Code / Program :-

```
# include <stdio.h>
int marh () {
        int row1, cols 1, rows 2, cols 2,
  print f ("Enter the number of rows and coloumns for motrix 1 \n");
 Scan f ("%d %d", $ rows1, cols 1);
 print f (" Enter the number of wows and columns of motrix 2 in");
 scan f ("%d %d" $ rows2, cols 2);
 f flush (stdin);
```

```
if (rows1 = = cols2){
       print ("multiplication will proceed \n");
else { print f (" multiplication is not present \n); }
        // Taking input for first matrix
   int i, j, a [row 1] [cols 1];
    for (i=0, ____ = row 1; i++) {
     for (j= 0; j = cols 1 ; j++) {
       print f (" Enter the values of matrix 1 \n");
       Scan f ("%d", $ a [i] [j]);
       f flush (stdin);
    }
     print f ("\n");
    }
    // Printing the first matrix
     int R, L;
    for (R = 0; R < rows 1; R++) {
       for (L = 0; L < cols 2; L++) {
           print f ("%d \t", a [R] [L]);
     }
      print f ("\n");
     }
    // Taking input for second matrix
       int m, n, b [rows2] [cols 2];
   for (m = 0;    m < rows 2 ; m++) {
     for (n = 0 ; n < cols 2 ; n++) {
         print f (" Enter the values for matrix 2 \n");
          Scan f ("%d", $ b [m] [n]);
        f flush (stdin);
       }
```

```
print f ("\n");
}
// printing the second matrix
int e, f;
 for (e=0; e< rows 2; C++){
   for (f=0; f < cols 2; f++){
     print f ("%d \t", b [e] [f]);
   }
   print f ("\n");
 }
 // initializing every element in product matrix to be 0
 int rows 3 = rows 1, cols 3 = cols 2; x, y;
   int ([rows 3] [cols 3];
     for (x=0; x< rows 3; x++){
       for (y=0; y< cols 3; y++){
         ([x][y] = 0;
     }
 }
 // multiplying the matrix
   int p, q, r;
 for (p=0; p< rows 1; p++) {
   for (q=0; q<cols2; q++) {
     for (r=0; r= cols 1; r++) {
       ([p][q] + = a [p][r]* b [r][q];
   }
 }
} // printing the product matrix
 int h, g;
 for (h=0, h< rows 3; h++){
   for (g=0; g<cols 3; g++) {printf ("%d \t", ([h][g]); }
   print f ("\n"); }
 }
```

Experiment No. → 10

* Aim :-
write a program to check whether a number is palindrome or not.

* Software Used :-
Code blocks GCC GNU Ming GW Compiler

* Theory :-
A palindrome number is a number that is same after reversing it. for eg: 121, 34543, etc.

* Code / Program :-

```
# include <stdio.h>
int main () {
        int n, r, sum = 0, ray;
         printf ("Enter the number");
          Scanf (" %d ", $n);
          Say = n
  while (n > 0) {
           r = n % 10;
          sum = (sum * 10) + r;
           n = n / 10;
     }
      if (sum = = say) {
             printf (" The entered number is a palindrome"); }
      else {
             printf ("The entered number is not a palindrome");
          }
```

Experiment No. - 11

* Aim:-
write a program to insert a substring inside a given string at a given index.

* Software used:-
Code block GCC GNU Ming GW compiler.

* Theory:-
concatenation is the process of appending one string to the end of another string. You concatenate strings by using the + operator. For string literals and string constants, concatenation occurs at compile time; no run-time concatenation occurs.

* Code / Program :-
```
# include <stdio.h>
void Insertstring (int p_index, const char* p_Givenstring,
          const char* p_substring);
  int main ( )
 {
   const char* Given String = "Happy Republic Day";
   const char* sub string = "74th";
   const int Index = 5;
   Insertstring (Index, Givenstring, substring);
   return 0;
 }
void Insert ~~String~~ (int p_index, const char* p_Givenstring,
          const char* p_substring)
 {
```

Date

Expt. No. Page No. 16

```
    for (int i = 0; i < strlen(p_Givenstring); i++)
{
   if (i == (p_index))
   {
      for (int j = 0; j < strlen(p_substring); j++)
      {
         printf("%c", p_substring[j]);
      }
   }
   else
   {
      printf("%c", p_Givenstring[i]);
   }
}
}
```

[illegible]

Teacher's Signature

Experiment No → 12

* Aim :-
To count the number of lines, words and character in a given line.

* Software Used :-
Code blocks Ming GW GNU GCC compiler 20.03.

* Theory :-
File Handling in C
In C we can perform 4 major file operations on files either text or binary :-
1. Creating a new file.
2. Opening an existing file.
3. closing a file
4. Reading from and writing information to a file when working with files we need to declare a pointer of type file. This declaration is needed for communication between the file and the program. for e.g.
FILE * fptr;

* Code / Program :-
```
    # include <stdio.h>
    # include <stdlib.h>

      int. main ()
      {
          FILE * fp
```

```
            fp = fopen(' new. text' 'r');
             if (!fp) {
        printf ('Error opening this file')s
    }
    int lines = 0, characters = 0, words = 0;
    int c ;
   while (c = getc (fp) != EOF) {
    ++ characters;
    if (c == '\n') {
         ++ lines;
    }
     if (c == '   ') {
          ++ words;
   }
 }
 printf (" lines: %d \n characters: %d \n words:
           %d \n", lines, characters, words);
    fclose (fp);
      return 0;
}
```

Experiment No → 13

* Aim :-

write a C program to calculate total and percentage marks of a student using structures.

* Software used :-

Code Blocks Ming GW GNU GCC compiler 20.03

* Theory :-

structures in C

In C programming, a struct (or structure) is a collection of (can be different types) under a single name.

Defining structures

Before we can create structure variable, you need to define its data type.

syntax :

```
struct structure name {
        data type member 1
        data type member 2
        ------
    }
```

* Code / Program :-

```
# include <stdio.h>
    struct student
            float marks [5];
             float total;
              float percentage;
        };
```

Date

Expt. No. Page No. 20

```
int main ( ) {
        struct student s ;
       int i ;

    print f ('Enter marks in 5 subjects');
     for (i=0; i<5 ; i++) {
         Scanf ("%f', &s . marks [i]);
          s.total = s.total + s. marks [i]
}
s. percentage =(s. total / 500)

print f (" marks in 5 subjects : ');
 for (i = 0; i<5 ; i++) {
    printf ("%.2f", s. marks [i]);

   }
   print f ("\n Total marks: %f \n", s. total);
   print f ("\n percentage: % f \n", s. persentage);
```

Teacher's Signature

Date

Expt. No.

Experiment No. → 14

* Aim:-

To sort a given array of numbers by insertion sort.

* Software Used:-

Code Blocks Ming GW GNU GCC compiler 20.03

* Theory:-

Insertion sort algorithm in C.

Insertion sort is a simple sorting algorithm that works similar to the way we sort playing cards in our hands. The array is virtually split into a sorted and an unsorted part. values from the unsorted part are picked and placed at correct position in the sorted part.

characterstics of Insertion sort:

(i) This algorithm is one of the simplest algorithm with simple implementation.

(ii) Insertion sort is efficient for small data values.

(iii) Insertion sort is adaptive in nature, i.e. it is adaptive appropriate for data sets which are already partially sorted.

* Code/Program:-

```
#include <math.h>
#include <stdio.h>
void insertion sort (int arr[], int n) {
    int i , key, j ;
    for (i=1 , i<n , i++){
        key = arr[i];
```

Teacher's Signature

Date

Expt. No. Page No. 22

```
            j = i-1 ;
            while (j > = 0  && arr [j] > key ) {
                   arr [j+1]  = arr[j]
                    j = j-1
            }
            arr [j+1] = key
        }
}
void print Array (int arr [ ], int n)
 {
     int i ;
     for (i = 0 ; i<n ; i++) {
         print f('%d', arr[i]);
       print f ("\n");
     }
 }
  int  main ( ) {
          int  arr [ ] = {12, 11, 13, 5, 6 };
          int n  = size of (arr) / size of (arr [0]);
          insertion sort (arr, n)
          print array (arr, n

          return 0 ;
   }
```

Teacher's Signature

Date

Expt. No.

Experiment No → 15

*** Aim :-**

To search an element from an array of elements using linear and binary search.

*** Software Used :-**

Ming GW GNU GCC compiler Code Blocks 20.03

*** Theory :-**

linear search in C

Item is in an array in random order so we have to find an item is to begin with the first pollution and compare it to target. if the item is at the some we will return the position of the current item. Otherwise we will return to the next position. This is called linear search.

Binary seach in C.

In Binary search, however cut down your search to half as soon as we find middle of the sorted list. The middle element is looked at to check if it is greater than or less than the value to be searched.

*** Code / Program :-**

```
#include <stdio.h>
 // Linear search in C.
  int search (int array [ ], int n, int x) {
        for (int i = 0 ; i<n ; i++) {
            if (array [i] == x) {
```

Teacher's Signature

Date

Expt. No. Page No. 24

```
                return i ;
            }
        }
    }
    int main ( ) {

            int array [] = {2, 6, 7, 13, 17, 21, 10}
            int n = 7
            int x = 17
            search (array , n , x);
            print f ("%d", i);
      }


      // Binary search
    # include < stdio.h >

     int binary search (int array [] , int x , int n){
             while ( low <= high){
                  int mid = low + chigh - low);
                  if (array [mid]== x ){
                       return mid ;
             }
             if (array [mid] < x ){
                     low = mid +1 ;
            }
             else {
              high = mid -1 ;
            }
```

[illegible]

Teacher's Signature

```
        }
          return 0
    }
      int main (){

          int array [] = {2, 6, 7, 13, 17, 21, 10}
          int n = 7
          int x = 17
        binary search (array, n, x);

    }
```