Command Line Codes

WRITE A PROGRAM FIBONACCI SERIES USING COMMAND LINE ARGUMENTS FOR TCS

```
#include<stdio.h>
#include<stdlib.h>

int main(int argc, char *argv[])

{

int n, first = 0, second = 1, next, c;

n = atoi(argv[1]);

printf("These are %d values in Fibonacci series are by PrepInsta:-\n",n);

for ( c = 0 ; c < n ; c++ )

{

if ( c <= 1 )

next = c;

else

{

next = first + second;

first = second;

second = next;

}

printf("%d\n",next);

}

return 0;

}
```

WRITE A PROGRAM TO SWAP TWO NUMBERS USING COMMAND LINE PROGRAMMING

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int main(int argc, char * argv[])
{
if(argc==1){
printf("No command line argument present, add them first");
return 0;
}
```

Command Line Codes

```
double firstNumber, secondNumber, temporaryVariable;
 firstNumber = atoi(argv[1]);

 secondNumber = atoi(argv[2]);

 temporaryVariable = firstNumber;

 firstNumber = secondNumber;

 secondNumber = temporaryVariable;

 printf("\nAfter swapping, firstNumber = %.2lf\n", firstNumber);

 printf("After swapping, secondNumber = %.2lf", secondNumber);

return 0;
}
```

STRING REVERSAL PROGRAM WITH COMMAND LINE PROGRAMMING

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main(int argc, char *argv[])
{
 int k;
 char temp;
 int i,j=0;
 int strsize = 0;
 for (i=1; i<argc; i++) {
 strsize += strlen(argv[i]);
 if (argc > i+1)
 strsize++;
 }
 char *cmdstring;
 cmdstring = malloc(strsize);
 cmdstring[0] = '\0';
 for (k=1; k<argc; k++) {
 strcat(cmdstring, argv[k]);
 if (argc > k+1)
 strcat(cmdstring, " ");
 }
 i = 0;
 j = strlen(cmdstring) - 1;
 while (i < j) {
 temp = cmdstring[i];
 cmdstring[i] = cmdstring[j];
 cmdstring[j] = temp;
 i++;
 j--;
 }
 printf("\nReverse string is :%s", cmdstring);
```

```
 return(0);

}
```

FIND GREATEST OF TWO NUMBER USING COMMAND LINE PROGRAMMING?

```
#include <stdio.h>

int main(int argc, char *argv[])

{

int c[10];

int i,temp,j,greatest;

j = 0;

for(i=1; i<argc; i++)

{

temp = atoi(argv[i]);

c[j] = temp;

j++;

}

greatest = c[0];

for (i = 0; i < 10; i++) {

if (c[i] > greatest) {

greatest = c[i];

}

}

printf("Greatest of ten numbers is %d", greatest);

return 0;

}
```

QUESTION: FIND THE LCM OF TWO NUMBERS USING COMMAND LINE LANGUAGE

```
#include <stdio.h>
#include <stdlib.h>
```

Command Line Codes

```
int main(int argc, char * argv[])
{
 int n1,n2,x,y;
 if (argc == 1 || argc > 3)
 {
 printf("Enter Two Number\r\n");
 exit(0);
 }
 x=atoi(argv[1]);
 y=atoi(argv[2]);
 n1 = x; n2 = y;
 while(n1!=n2){
 if(n1>n2)
 n1=n1-n2;
 else
 n2=n2-n1;
 }
 printf("L.C.M of %d & %d = %d \r\n",x,y,x*y/n1);
 return 0;
}
```

FIND THE AVERAGE OF TWO NUMBERS USING COMMAND LINE LANGUAGE

```
#include <stdio.h>

int main(int argc, char * argv[])
{
 int sum = 0,i = 1,count = 0;
 if(argc == 1)
 {
 printf("Enter the number \n");
 exit(1);
 }
 count = argc - 1;
 while (i <= count )
 {
 sum += atoi (argv[i]) ;
 i++;
 }
 printf("Avg of the numbers.%d\n", sum/count);
}
```

FIND THE SUM OF THE DIGITS OF A NUMBER BY COMMAND LINE ARGUMENTS

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char * argv[])
{
    long num, temp, digit, sum = 0;
    if(argc == 1 ||  argc > 2)
    {
         printf("Enter the number\n");
```

```
        exit(1);
    }
    num = atoi (argv[1]) ;
    temp = num;
    while (num > 0)
    {
        digit = num % 10;
        sum  = sum + digit;
        num /= 10;
    }
    printf("Sum of the digits of %ld = %ld\n", temp, sum);
}
```

SAMPLE PROGRAM TO PRINT ALL INTEGERS USING COMMAND LINE ARGUMENTS

```
#include <stdio.h>

int main(int argc, char *argv[])
{
int a,b;
int i;
if(argc<2)
{
printf("please use \"prg_name value1 value2 ... \"\n");
return -1;
}

for(i=1; i<argc; i++)
{
printf("arg[%2d]: %d\n",i,atoi(argv[i]));
}


return 0;
}
```

TCS COMMAND LINE ARGUMENT PROGRAM FOR FACTORIAL OF A NON-NEGATIVE INTEGER

```
#include <stdio.h> // for printf
#include <stdlib.h> // for function atoi() for converting string into int
// Function to return fact value of n
int fact(int n)
{
 if (n == 0)
 return 1;
 else {
 int ans = 1;
 int i;
 for (i = 1; i <= n; i++) {
 ans = ans * i;
 }
 return ans;
 }
}
// argc tells the number of arguments
```

Command Line Codes

```
// provided+1 +1 for file.exe
// char *argv[] is used to store the
// command line arguments in the string format
int main(int argc, char* argv[])
{
 // means only one argument exist that is file.exe
 if (argc == 1) {
 printf("No command line argument exist Please provide them first \n");
 return 0;
 } else {
 int i, n, ans;
 // actual arguments starts from index 1 to (argc-1)
 for (i = 1; i < argc; i++) {
 // function of stdlib.h to convert string
 // into int using atoi() function
 n = atoi(argv[i]);

 // since we got the value of n as usual of
 // input now perform operations
 // on number which you have required

 // get ans from function
 ans = fact(n);

 // print answer using stdio.h library's printf() function
 printf("%d\n", ans);
 }
 return 0;
 }
}
```

CALCULATE LENGTH OF THE HYPOTENUSE OF RIGHT ANGLED TRIANGLE

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{

if(argc<2)
{
printf("please use \"prg_name value1 value2 ... \"\n");
return -1;
}

int a,b,side1,side2,side3;
a=atoi(argv[1]);
b=atoi(argv[2]);
side1=pow(a,2);
side2=pow(b,2);
side3=sqrt((side1+side2));
printf("the hypotenuse is %d",side3);
return 0;
```

```
}
```

WRITE A C PROGRAM TO CONVERT BINARY TO DECIMAL USING COMMAND LINE ARGUMENTS

```
#include<stdio.h>
int main(int argc, char *argv[]){
int num,binary,decimal=0,rem,base=1;
num=atoi(argv[1]);
binary=num;
while(num>0){
rem=num%2;
decimal+=rem*base;
num=num/10;
base=base*2;
}
printf("%d",decimal);
return 0;
}
```

WRITE A C PROGRAM TO CHECK WHETHER GIVEN NO. IS PALINDROME OR NOT USING COMMAND LINE ARGUMENTS

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char* argv[])
{
 int num=atoi(argv[1]);
 if(isPalindrome(num))
 printf("Palindrome");
 else
 printf("Not Palindrome");

 return 0;
}
int isPalindrome(int n)
{
 int m=n;
 int rev=0;
 while(m!=0)
 {
 rev=(rev*10) + (m%10);
 m=m/10;
 }
 if(rev==n)
 return 1;
 else
 return 0;
}
```

Write a C program that will find the sum of all prime numbers in a given range.The range will be specified as command line parameters. The first command line parameter, N1 which is a positive integer, will contain the lower bound of the range. The second command line parameter N2, which is also a positive integer will the upper bound of the range. The program should consider all the prime numbers within the range, excluding the upper and lower

bound. Print the output in integer format to stdout. Other than the integer number, no other extra information should be printed to stdout.

```
#include<stdio.h>
int main(int argc,char *argv[])
{
int N1,N2,i,j,sum=0,count,lower,upper;
if(argc!=3)
 exit(0);
N1=atoi(argv[1]);
lower=N1+1;
N2=atoi(argv[2]);
upper=N2;
 for(i=lower;i<upper;i++)
 {
 count=1;
 for(j=2;j<=i/2;j++)
 {
 if(i%j==0)
 {
 count++;
 }
 }
 if(count==1)
 {
 sum=sum+i;
 }
 }
 printf("%d",sum);
return 0;
}
```

COMMAND LINE PROGRAM FOR CHECKING PALINDROME (STRING)

```
#include <stdio.h>

#include <string.h>

void isPalindrome(char str[])

{

int l = 0;

int h = strlen(str) - 1;

while (h > l)

{

if (str[l++] != str[h--])

{

printf("%s is Not Palindromen", str);
```

Command Line Codes

```
return;

}

}

printf("%s is palindromen", str);

}

int main(int argc, char *argv[])

{

int i,k;

int strsize = 0;

for (i=1; i<argc; i++) {

strsize += strlen(argv[i]);

if (argc > i+1)

strsize++;

}

char *cmdstring;

cmdstring = malloc(strsize);

cmdstring[0] = '\0';

for (k=1; k<argc; k++) {

strcat(cmdstring, argv[k]);

if (argc > k+1)

strcat(cmdstring, " ");

}

isPalindrome(cmdstring);

}
```

SORT AN ARRAY INTO TWO HALVES, ONE HALF ASCENDING AND SECOND HALF DESCENDING COMMAND LINE LANGUAGE

```
// C++ program to print first half in
// ascending order and the second half
```

Command Line Codes

```
// in descending order
#include <bits/stdc++.h>
using namespace std;

// function to print half of the array in
// ascending order and the other half in
// descending order
void printOrder(int arr[], int n)
{
 // sorting the array
 sort(arr, arr + n);

 // printing first half in ascending
 // order
 for (int i = 0; i < n / 2; i++)
 cout << arr[i] << " ";

 // printing second half in descending
 // order
 for (int j = n - 1; j >= n / 2; j--)
 cout << arr[j] << " ";
}

// driver code
int main()
{
 int arr[] = { 5, 4, 6, 2, 1, 3, 8, 9, 7 };
 int n = sizeof(arr) / sizeof(arr[0]);
 printOrder(arr, n);

 return 0;
}
```

REVERSE DIGITS OF A NUMBER USING TCS COMMAND LINE ARGUMENTS

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
 if(argc==1)
 {
 printf("No Arguments");
 return 0;
 }
 else
 {
 int n,reverseNumber,temp,rem;
 n=atoi(argv[1]);
 temp=n;
 reverseNumber=0;
 while(temp)
 {
 rem=temp%10;
```

Command Line Codes

```
 reverseNumber=reverseNumber*10+rem;
 temp=temp/10;
 }
 printf("%d",reverseNumber);
 return 0;
 }
}
```

DECIMAL TO BINARY USING COMMAND LINE ARGUMENTS

```
#include<stdio.h>
#include<stdlib.h>
 int main(int argc, char *argv[])
 {
 if(argc==1)
 {
 printf("No Arguments ");
 return 0;
 }
 else
 {
 int n;

 n=atoi(argv[1]);
 int binaryN[64];
 int i=0;int j;
 while(n>0)
 {
 //storing in binary array remainder of number
 binaryN[i]=n%2;
 n=n/2;
 i++;
 }
 //printing reverse array
 while(i)
 {
 printf("%d",binaryN[--i]);
 }

return 0;
 }
 }
```

THE SQUARE ROOT OF A PRIME NUMBER BY CHECKING FIRST IF IT IS A PRIME NUMBER

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<math.h>
bool isPrime(int n)
{
if(n<2)
return false;
int i;
for(i=2;i*i<=n;i++)
```

Command Line Codes

```
{
if(n%i==0)
return false;
}
return true;
}
int main(int argc, char *argv[])
{
if(argc==1)
{
printf("No arguments");
return 0;
}
else
{
int n;
n=atoi(argv[1]);
float sq=0;
if(isPrime(n))
{
sq=sqrt(n);
printf("%.2f",sq);
}
else
printf("%.2f",sq);
return 0;
}
}
```

C PROGRAM TO FIND ARMSTRONG NUMBER USING COMMAND LINE ARGUMENTS

```
#include<stdio.h>
int main(int argc, char * argv[])
{
int num,temp,arms=0,rem;
if (argc!= 2)
{
 printf("Enter the number:\n");
 scanf("%d",&num);
}
else
{
 num = atoi(argv[1]);
}
 temp=num;
while(num>0)
{
 rem=num%10;
 arms=arms+rem*rem*rem;
 num=num/10;
}
if(temp==arms)
{
```

```
 printf(" \n%d is an Armstrong number",temp);
}
else
{
 printf("\n%d is not an Armstrong number",temp);
}
return 0;
}
```

TCS COMMAND LINE PROGRAM FOR BINARY TO OCTAL

```
#include<stdio.h>
void main(int argc,char *argv[])
{
 long int n,r,c,b=1,s=0;
 n=atoi(argv[1]);
 c=n;
 while(c!=0)
 {
 r=c%10;
 s=s+r*b;
 c=c/10;
 b=b*2;
 }
 printf("%lo",s);
 getch();
}
```

TCS COMMAND LINE PROGRAM FOR DECIMAL TO OCTAL

```
#include<stdio.h>
int main(int argc,char *argv[])
{
 int n,s=0,b=1,r;
 n=atoi(argv[1]);
 int c=n;
 while(c>0)
 {
 r=c%8;
 s=s+r*b;
 c=c/8;
 b=b*10;
 }
 printf("%d",s);
 getch();
}
```

COMMAND LINE PROGRAM TO CHECK LEAP YEAR

```
#include<stdio.h>
void main(int argc,char *argv[])
{
 int n;
 n=atoi(argv[1]);
 if(n%4==0)
 {
```

Command Line Codes

```
 if(n%100==0)
 {
 if(n%400==0)

 printf("Leap Year");
 else
 printf("Not Leap Year");
 }
 else
 printf("Leap Year");
 }
 else
 printf("Not Leap Year");
 getch();

}
```