TCS Previous Papers
www.prepinsta.com

Coding via Command Line

TCS Previous Year Papers and Study materials

Theory

For TCS Command Line Arguments Programs let us consider this, if you wanted to write a basic C program then you would've written a main function that would've looked like –

```
int main(){
// some code in Command Line Arguments Questions for TCS C Compiler
}}
```

However in command line arguments we write like this –

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

1. **argc** – It is know as Argument Count and as clear from the name it stores the Count of number of Arguments.
2. **argv[] –** Pointer, contains location of all the values(arguments).
3. ***argv[] –** Array of values of all the arguments.
4. They are parameters/arguments supplied to the program when it is invoked.

Thus, now we have two things for TCS C Compiler

1. Total Count of number of Arguments.
2. All the values/pointer location of arguments stored in an array.

Now, you will need one more thing i.e. atoi();

- **atoi(); –** Converts string into int and atoi(argv[i]); will give the value of argument at ith location in int type format.

Now you can use an int val = atoi(argv[i]); to store the value and then print it with printf(); function.

To pass command line arguments, we typically define main() with two arguments : first argument is the number of command line arguments and second is list of command-line arguments.

```
int main(int argc, char *argv[]) { /* ... */ }
```

or

```
int main(int argc, char **argv[]) { /* ... */ }
```

Quick Facts for TCS Command Line Arguments Programs

argv[0] contains the default value, not the input value so –

- All for loops must start from i = 1.
- You must use the following condition

```
if(argc == 1){
// do nothing since, there are no arguments, maybe ask for arguments?
}else{
// code to apply logic and print values.
}
```

- provided+1 +1 for file.exe
- argv[argc] is a NULL pointer.
- argv[0] holds the name of the program.
- argv[1] points to the first command line argument and argv[n] points last argument.

Watch these videos here -

https://www.youtube.com/watch?v=q7A-RaSLSrY

https://www.youtube.com/watch?v=fzS4ZvwTj00

https://www.youtube.com/watch?v=HdNa_nEx_N0

Command Line Arguments Questions for TCS Rules for Coding Section Steps:

There is only one question for 20 minutes.

1. It has 10 attempts(We can compile only 10 times).
2. We must start our code from the scratch.
3. The coding platform is divided into two, one for writing the code and other for output. We should write the whole program.
4. We can’t use any input functions like scanf(), getch(), getchar().
5. The input to be provided should be read as command line arguments.

We must only print exact output in Command Line Arguments Questions for TCS Procedure –

1. Output must not be re-framed by extra words.
2. If there is any error, the error will be shown in the output dialog box.
3. The errors are clearly mentioned.
4. If there are no errors, a message like “compiled successfully” will be printed.
5. Along with that they will mention four test cases are ‘passed’ or maybe ‘failed’.
6. They are indicated like private and public test cases. They have not mentioned what is the test case, which is difficult to understand in TCS Command Line Arguments Programs.

Dont Compile again and again since compiler takes 25 seconds and each time you compile 25 seconds will become lesser in the time you have to code.

Command Line Program for Odd Even

```
#include <stdio.h>
int main(int argc, char *argv[])
{
int number;
number = atol(argv[1]);
if(number % 2 == 0)
printf(“%d is even.”, number);
else
printf(“%d is odd.”, number);
return 0;
}
```

Command Line Program to Convert Binary to Octal

This is a very smart program very short and quick method –

```
#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();
}
```

Command Line Program to Convert Decimal to Octal

This is very smart short and quick program –

```
#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 if a year is Leap Year or Not

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

}
```

Command Line Program to find Area of Circle

Write a C program to find the area of a circle with radius provided.
The value of radius positive integer passed to the program as the first command line parameter. Write the output to stdout formatted as a floating point number rounded to EXACTLY 2 decimal precision WITHOUT any other additional text.
Scientific format(such as 1.00E+5) should NOT be used while printing the output.
You may assume that the inputs will be such that the output will not exceed the largest possible real number that can be stored in a float type variable.

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char * argv[])
{
 if(argc==1)
 {
```

```
 printf("No arguments");
 return 0;
 }
 else
 {
 int radius;
 float pi=3.14;
 float area;
 radius=atoi(argv[1]);
 area=pi*radius*radius;
 printf("%.2f",area);
 return 0;
 }
}
```

Write a C program to find the area of a circle with radius provided.

The value of radius positive integer passed to the program as the first command line parameter. Write the output to stdout formatted as a floating point number rounded to EXACTLY 2 decimal precision WITHOUT any other additional text.

Scientific format(such as 1.00E+5) should NOT be used while printing the output.

Command Line Program to find Area of Traingle

```
#include <stdio.h>

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

{
```

```
         int base, height;

        float area;

         base = atol(argv[1]);

        height = atol(argv[2]);

        area = base*height/2;

        printf("%f",area);

}
```

Command Line Program to Check if a Number is Palindrome or Not

Palindrome Number Command Line Programming

```

#include <stdio.h>

int main(int argc, char *argv[])
{
          int num, reverse_num=0,remainder,temp;
          num = atol(argv[1]);
          temp=num;
            while(temp!=0)
   {
          remainder=temp%10;
          reverse_num=reverse_num*10+remainder;
          temp/=10;
   }
   if(reverse_num==num)
          printf("%d is a palindrome number",num);
   else
          printf("%d is not a palindrome number",num);
   return 0;
}
```

Command Line Program to Check if a String is Palindrome or Not

```
#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);

        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);
}
```

Command Line Program to Check if a Number is Prime or Not

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

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

{

         int n, i, flag = 0;

         n = atol(argv[1]);

        for(i=2; i<=n/2; ++i)

   {

                if(n%i==0)

        {

           flag=1;

           break;

        }

   }

   if (flag==0)

                printf("%d is a prime number.",n);

   else

                printf("%d is not a prime number.",n);
```

```
  return 0;

}
```

Command Line Program to Reverse a Number

```
#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;

 reverseNumber=reverseNumber*10+rem;

 temp=temp/10;

 }

 printf("%d",reverseNumber);

 return 0;

 }

}
```

Calculate Square Root without using Math.h sqrt Function

```
#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]);
 float i=0.00;
 while(i*i<=n)
 {
 i=i+0.001;
 }
 i=i-0.001;
 printf("%.2f",i);
 }
}
```

Square Root of Prime Number using Command Line Argument

```
#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++)

{

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;
}
}
```

Nth Fibonacci Number

```
#include<stdio.h>
#include<stdlib.h>
 int fib(int n)
```

```
{
int a=0,b=1,c,i;
if(n==0) return a;
for(i=2;i<=n;i++)
{
c=a+b;
a=b;
b=c;
}
return b;
}
int main(int argc, char * argv[])
{
if(argc==1)
{
printf("No arguments");
return 0;
}
else
{
int n;
```

```
 n=atoi(argv[1]);

 printf("%d",fib(n));

 return 0;

 }

 }
```

Decimal to Binary

```
#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;

 }

 }
```

Binary to decimal using Command Line Language

```
#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;
}
```

Sum of all the digits of a number using Command Line Language

```
#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);
}
```

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);
}
```

Nth Fibonacci Number

```
#include<stdio.h>

#include<stdlib.h>

 int fib(int n)

 {

 int a=0,b=1,c,i;

 if(n==0) return a;

 for(i=2;i<=n;i++)

 {
```

```
c=a+b;

a=b;

b=c;

}

return b;

}

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

{

if(argc==1)

{

printf("No arguments");

return 0;

}

else

{

int n;

n=atoi(argv[1]);

printf("%d",fib(n));

return 0;

}

}
```

LCM of Two Numbers using Command Line Language

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

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;
}
```

Command Line Programming For Greatest of Two numbers

```
#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;

}
```

Tcs Command Line Program for String Reversal

```
#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);

}
```

Command Line Program for Swapping two Numbers

```
#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;

}


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;

}
```

Fibonacci Series Program – Using Command Line Arguments

```
#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;
```

```
}
```

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
// 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;
 }
}
// Program to print all value of
 // command line argument
 // once we get the value from command
 // line we can use them to solve our problem
 #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;
 }
```