CENTRAL CALCUTTA POLYTECHNIC

21, Convent Road, Philips, Sealdah, Kolkata, West Bengal 700014

DEPT. : COMPUTER SCIENCE AND TECHNOLOGY

- NAME : ASIM BERA
- ROLL : DCCPCSTS3
- NO : 10005504
- SUBJECT : PROGRAMMING IN C
- SESSION : 2020 - 21

Contents

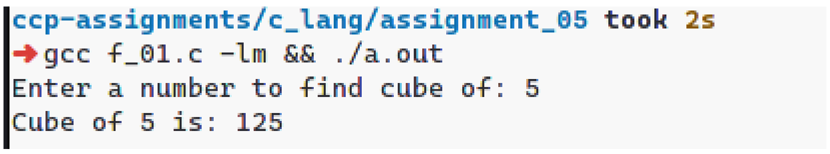
1 Functions

1.1 Write a C program to find cube of any number using function.

Source Code :

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

int cubeOf(int val)
{
  int res = pow(val, 3);
  return res;
}

int main()
{
  int a, b;
  printf("Enter a number to find cube of: ");
  scanf("%d", &a);
  b = cubeOf(a);
  printf("Cube of %d is: %d\n", a, b);
  return 0;
}
```

Program Output :



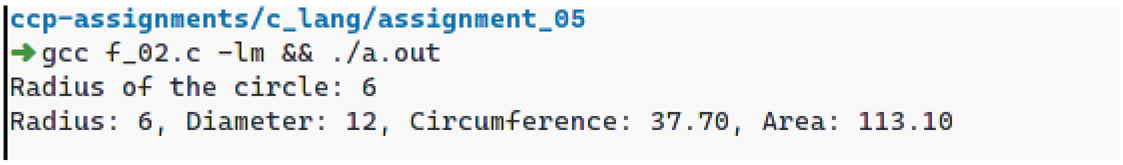
```
ccp-assignments/c_lang/assignment_05 took 2s
➜ gcc f_01.c -lm && ./a.out
Enter a number to find cube of: 5
Cube of 5 is: 125
```

1.2 Write a C program to find diameter, circumference and area of circle using functions.

Source Code :

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

int diameterOf(int val)
{
  return 2 * val;
}
double circumferenceOf(int val)
{
  return 2 * M_PI * val;
}
double areaOf(int val)
{
  return M_PI * pow(val, 2);
}

int main()
{
  int a;
  printf("Radius of the circle: ");
  scanf("%d", &a);
  printf("Radius: %d, Diameter: %d, Circumference: %.2f, Area: %.2f\n", a, diameterOf(a),
  ↪   circumferenceOf(a), areaOf(a));
  return 0;
}
```

Program Output :



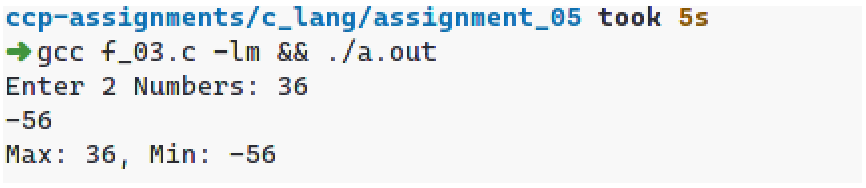
```
ccp-assignments/c_lang/assignment_05
➜ gcc f_02.c -lm && ./a.out
Radius of the circle: 6
Radius: 6, Diameter: 12, Circumference: 37.70, Area: 113.10
```

1.3 Write a C program to find maximum and minimum between two numbers using functions.

Source Code :

```c
#include <stdio.h>

int minMaxOf(int a, int b, int flag)
{
  // flag: 0 = Min, 1 = Max
  if (flag)
    return a > b ? a : b;
  return a > b ? b : a;
}

int main()
{
  int a, b;
  printf("Enter 2 Numbers: ");
  scanf("%d", &a);
  scanf("%d", &b);
  printf("Max: %d, Min: %d\n", minMaxOf(a, b, 1), minMaxOf(a, b, 0));
}
```

Program Output :



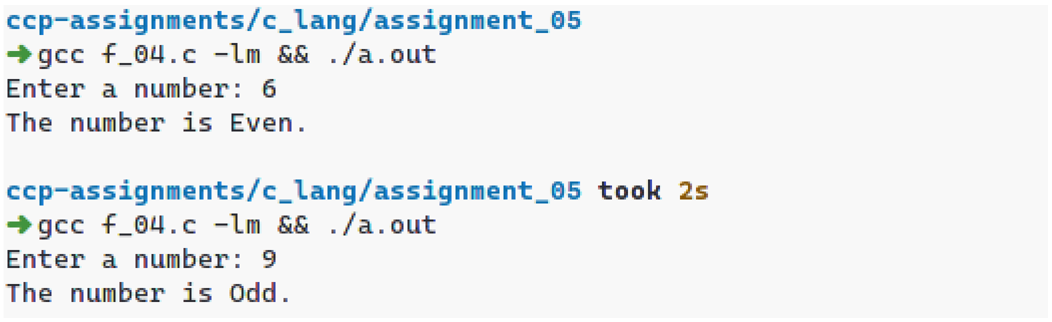
```
ccp-assignments/c_lang/assignment_05 took 5s
➜gcc f_03.c -lm && ./a.out
Enter 2 Numbers: 36
-56
Max: 36, Min: -56
```

1.4 Write a C program to check whether a number is even or odd using functions.

Source Code :

```c
#include <stdio.h>

int isEvenOrOdd(int val)
{
  // even = 1, odd = 0
  return val % 2 == 0 ? 1 : 0;
}

int main()
{
  int a;
  printf("Enter a number: ");
  scanf("%d", &a);

  printf("The number is %s.\n", isEvenOrOdd(a) ? "Even" : "Odd");

  return 0;
}
```

Program Output :



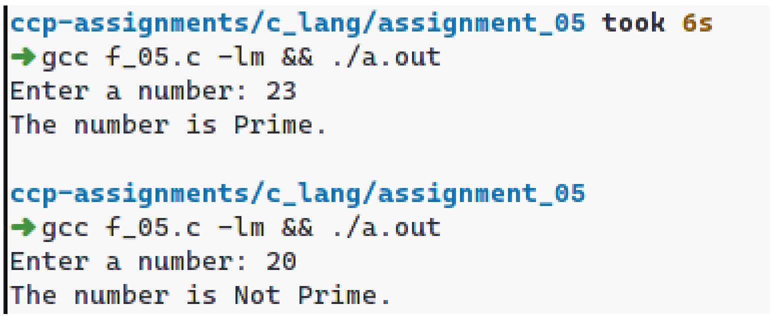
```
ccp-assignments/c_lang/assignment_05
➜gcc f_04.c -lm && ./a.out
Enter a number: 6
The number is Even.

ccp-assignments/c_lang/assignment_05 took 2s
➜gcc f_04.c -lm && ./a.out
Enter a number: 9
The number is Odd.
```

1.5 Write a C program to check whether a number is prime or not using function.

Source Code :

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

// check using trial division method
// prime = 1, not prime = 0
int IsPrime(int val)
{
  int sqrtOfVal = sqrt(val), prime = 1;
  for (int i = 2; i <= sqrtOfVal; i++)
  {
    if (val % i == 0)
    {
      prime = 0;
      break;
    }
  }

  return prime;
}

int main()
{
  int a;
  printf("Enter a number: ");
  scanf("%d", &a);

  printf("The number is %s.\n", IsPrime(a) ? "Prime" : "Not Prime");

  return 0;
}
```

Program Output :



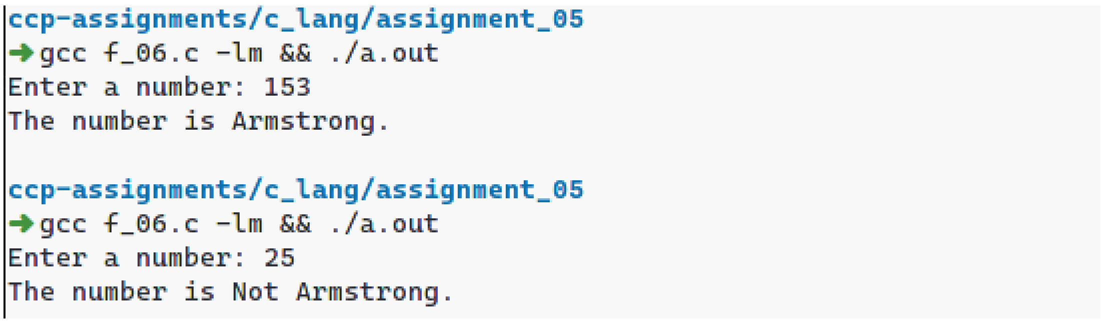
```
ccp-assignments/c_lang/assignment_05 took 6s
➔gcc f_05.c -lm && ./a.out
Enter a number: 23
The number is Prime.

ccp-assignments/c_lang/assignment_05
➔gcc f_05.c -lm && ./a.out
Enter a number: 20
The number is Not Prime.
```

1.6 Write a program to check whether a number is an Armstrong number or not using function.

Source Code :

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

int isArmstrong(int val)
{
  // int digits = 0;
  int i = val, total = 0;
  while (i > 0)
  {
    // digits++;
    total += pow(i % 10, 3);
    i = i / 10;
  }
  return total == val ? 1 : 0;
}

int main()
{
  int a;
  printf("Enter a number: ");
  scanf("%d", &a);

  printf("The number is %s.\n", isArmstrong(a) ? "Armstrong" : "Not Armstrong");

  return 0;
}
```

Program Output :



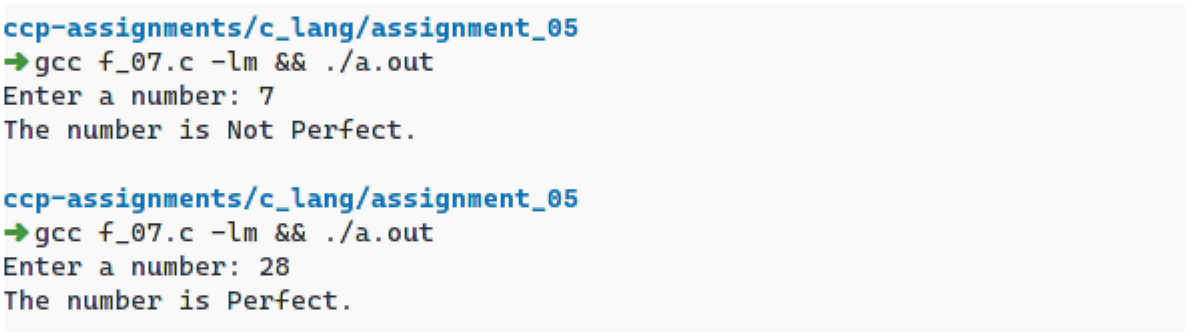
```
ccp-assignments/c_lang/assignment_05
➜gcc f_06.c -lm && ./a.out
Enter a number: 153
The number is Armstrong.

ccp-assignments/c_lang/assignment_05
➜gcc f_06.c -lm && ./a.out
Enter a number: 25
The number is Not Armstrong.
```

1.7 WAP to check a number is perfect number or not using functions.

Source Code :

```c
#include <stdio.h>

int isPerfect(int val)
{
  int total = 0;
  for (int i = 1; i < val; i++)
  {
    if (val % i == 0)
      total += i;
  }

  return total == val ? 1 : 0;
}

int main()
{
  int a;
  printf("Enter a number: ");
  scanf("%d", &a);

  printf("The number is %s.\n", isPerfect(a) ? "Perfect" : "Not Perfect");

  return 0;
}
```

Program Output :



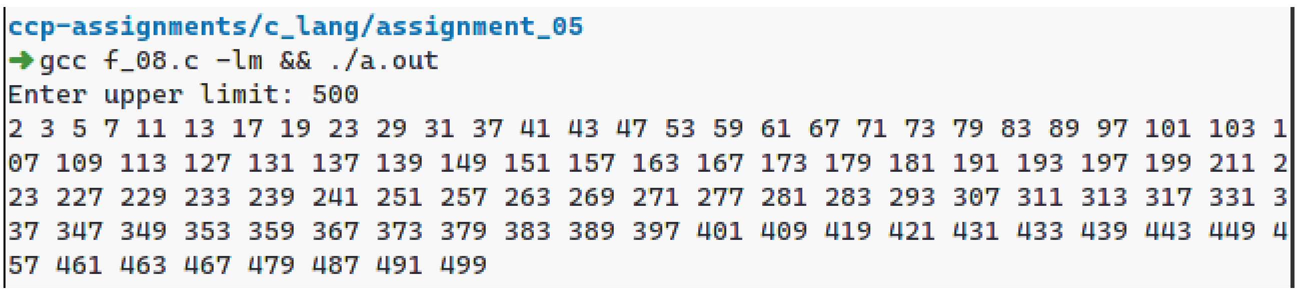
```
ccp-assignments/c_lang/assignment_05
➜ gcc f_07.c -lm && ./a.out
Enter a number: 7
The number is Not Perfect.

ccp-assignments/c_lang/assignment_05
➜ gcc f_07.c -lm && ./a.out
Enter a number: 28
The number is Perfect.
```

1.8 Write a C program to find all prime numbers between given interval using functions.

Source Code :

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

// Prime checking using Sieve of Eratosthenes
void enumeratePrime(int lim)
{
  int flags[lim];
  for (int i = 0; i < lim; i++)
    flags[i] = 1;
  for (int i = 2; i < sqrt(lim); i++)
  {
    if (flags[i])
    {
      int j = pow(i, 2);
      while (j < lim)
      {
        flags[j] = 0;
        j += i;
      }
    }
  }

  for (int i = 2; i < lim; i++)
  {
    if (flags[i])
    {
      printf("%d ", i);
    }
  }
}

int main()
{
  int a;
  printf("Enter upper limit: ");
  scanf("%d", &a);

  enumeratePrime(a);

  return 0;
}
```

Program Output :



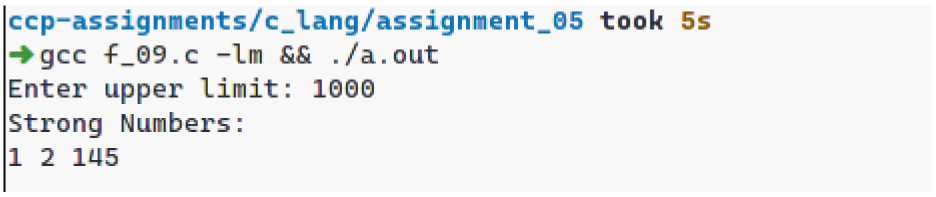
```
ccp-assignments/c_lang/assignment_05
➜gcc f_08.c -lm && ./a.out
Enter upper limit: 500
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 1
07 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 2
23 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 3
37 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 4
57 461 463 467 479 487 491 499
```

1.9 Write a C program to print all strong numbers between given interval using functions.

Source Code :

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

int isStrongNumber(int val)
{
  int i = val, total = 0;
  while (i > 0)
  {
    // using tgamma as factorial alternative
    // from math.h
    total += tgamma((i % 10) + 1);
    i = i / 10;
  }
  return total == val ? 1 : 0;
}

void enumerateStrong(int lim)
{
  printf("Strong Numbers: \n");
  for (int i = 1; i <= lim; i++)
  {
    if (isStrongNumber(i))
    {
      printf("%d ", i);
    }
  }
}

int main()
{
  int a;
  printf("Enter upper limit: ");
  scanf("%d", &a);

  enumerateStrong(a);

  return 0;
}
```

Program Output :



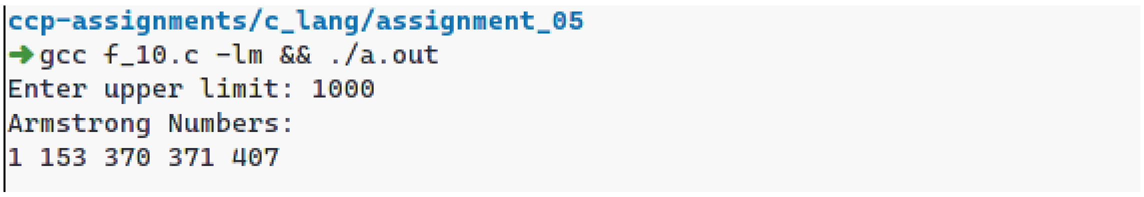
```
ccp-assignments/c_lang/assignment_05 took 5s
➜gcc f_09.c -lm && ./a.out
Enter upper limit: 1000
Strong Numbers:
1 2 145
```

1.10 Write a C program to print all Armstrong numbers between given interval using functions.

Source Code :

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

int isArmstrongNumber(int val)
{
  int i = val, total = 0;
  while (i > 0)
  {
    total += pow(i % 10, 3);
    i = i / 10;
  }
  return total == val ? 1 : 0;
}

void enumerateArmstrong(int lim)
{
  printf("Armstrong Numbers: \n");
  for (int i = 1; i <= lim; i++)
  {
    if (isArmstrongNumber(i))
    {
      printf("%d ", i);
    }
  }
}

int main()
{
  int a;
  printf("Enter upper limit: ");
  scanf("%d", &a);

  enumerateArmstrong(a);

  return 0;
}
```

Program Output :



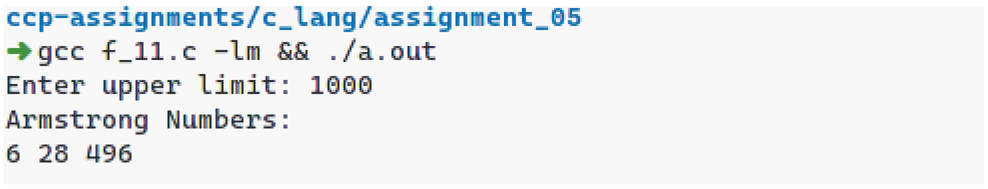
```
ccp-assignments/c_lang/assignment_05
➜gcc f_10.c -lm && ./a.out
Enter upper limit: 1000
Armstrong Numbers:
1 153 370 371 407
```

1.11 Write a C program to print all perfect numbers between given interval using functions.

Source Code :

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

int isPerfectNumber(int val)
{
  int total = 0;
  for (int i = 1; i < val; i++)
  {
    if (val % i == 0)
      total += i;
  }

  return total == val ? 1 : 0;
}

void enumeratePerfect(int lim)
{
  printf("Armstrong Numbers: \n");
  for (int i = 1; i <= lim; i++)
  {
    if (isPerfectNumber(i))
    {
      printf("%d ", i);
    }
  }
}

int main()
{
  int a;
  printf("Enter upper limit: ");
  scanf("%d", &a);

  enumeratePerfect(a);

  return 0;
}
```

Program Output :



```
ccp-assignments/c_lang/assignment_05
➜gcc f_11.c -lm && ./a.out
Enter upper limit: 1000
Armstrong Numbers:
6 28 496
```

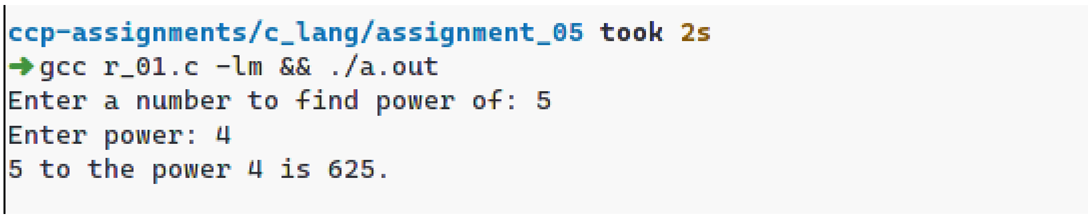
2 Recursion

2.1 Write a C program to find power of any number using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Recurring function to find power of given integer.
 * @param val current value
 * @param init initial integer to calculate on
 * @param pow decreasing power
 */
int powOf(int val, int init, int pow)
{
  if (pow == 0)
    return val;
  return powOf(val * init, init, pow - 1);
}

int main()
{
  int a, b;
  printf("Enter a number to find power of: ");
  scanf("%d", &a);
  printf("Enter power: ");
  scanf("%d", &b);

  printf("%d to the power %d is %d.\n", a, b, powOf(1, a, b));
  return 0;
}
```

Program Output :



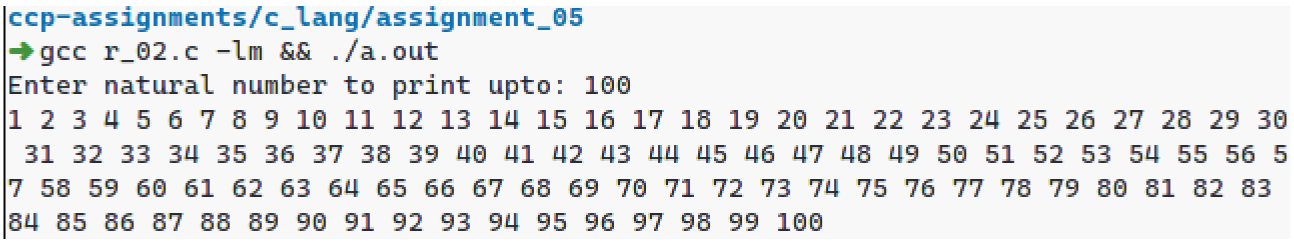
```
ccp-assignments/c_lang/assignment_05 took 2s
➜gcc r_01.c -lm && ./a.out
Enter a number to find power of: 5
Enter power: 4
5 to the power 4 is 625.
```

2.2 Write a C program to print all natural numbers between 1 to n using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Print all numbers between 1 and [lim]. Both included.
 * @param lim upper limit
 * @param cur current number
*/
void enumerateNaturals(int lim, int cur)
{
  if (cur > lim)
    return;
  printf("%d ", cur);
  return enumerateNaturals(lim, cur + 1);
}

int main()
{
  int a;
  printf("Enter natural number to print upto: ");
  scanf("%d", &a);
  enumerateNaturals(a, 1);
  return 0;
}
```

Program Output :



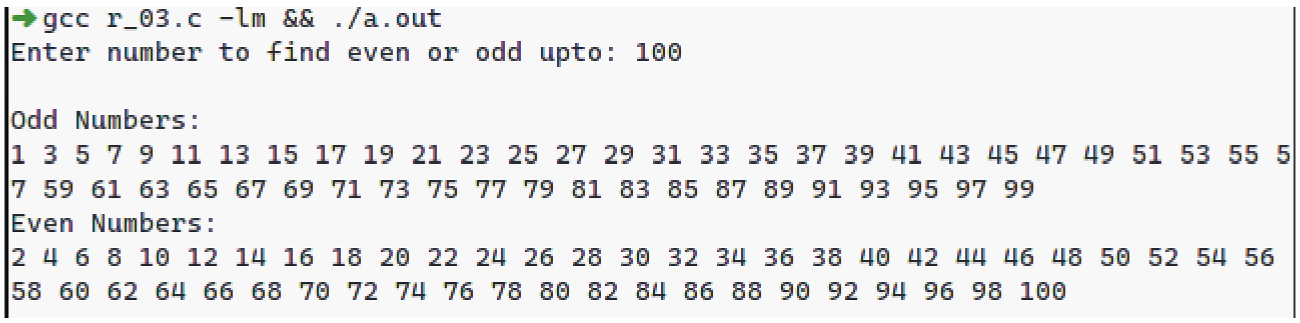
```
ccp-assignments/c_lang/assignment_05
➜gcc r_02.c -lm && ./a.out
Enter natural number to print upto: 100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 5
7 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

2.3 Write a C program to print all even or odd numbers in given range using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Print all even or odd numbers between 1 and [lim]. Both included.
 * @param lim upper limit
 * @param cur current number
 * @param type flag for odd = 0, even = 1
*/
void enumerateEvenOrOdd(int lim, int cur, int type)
{
  if (cur > lim)
    return;
  if (type)
  {
    if (cur % 2 == 0)
      printf("%d ", cur);
  }
  else
  {
    if (cur % 2 != 0)
      printf("%d ", cur);
  }
  return enumerateEvenOrOdd(lim, cur + 1, type);
}

int main()
{
  int a;
  printf("Enter number to find even or odd upto: ");
  scanf("%d", &a);
  printf("\nOdd Numbers: \n");
  enumerateEvenOrOdd(a, 1, 0);
  printf("\nEven Numbers: \n");
  enumerateEvenOrOdd(a, 1, 1);
  return 0;
}
```

Program Output :



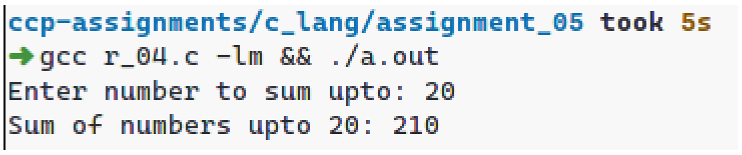
```
➜ gcc r_03.c -lm && ./a.out
Enter number to find even or odd upto: 100

Odd Numbers:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 5
7 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
Even Numbers:
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56
58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100
```

2.4 Write a C program to find sum of all natural numbers between 1 to n using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Find sum upto integer [lim].
 * @param lim upper limit
 * @param cur current integer
 * @param sum sum of numbers upto [cur]
*/
int sumUpto(int lim, int cur, int sum)
{
  if (cur > lim)
    return sum;
  return sumUpto(lim, cur + 1, sum + cur);
}

int main()
{
  int a;
  printf("Enter number to sum upto: ");
  scanf("%d", &a);
  printf("Sum of numbers upto %d: %d\n", a, sumUpto(a, 1, 0));
  return 0;
}
```

Program Output :



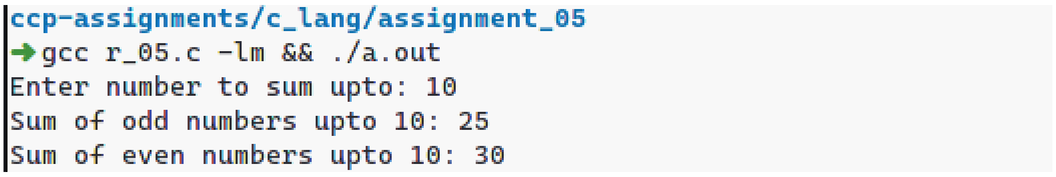
```
ccp-assignments/c_lang/assignment_05 took 5s
➜gcc r_04.c -lm && ./a.out
Enter number to sum upto: 20
Sum of numbers upto 20: 210
```

2.5 Write a C program to find sum of all even or odd numbers in given range using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Find sum upto integer [lim].
 * @param lim upper limit
 * @param cur current integer
 * @param sum sum of numbers upto [cur]
 * @param type flag of odd = 0, even = 1
*/
int sumUpto(int lim, int cur, int sum, int type)
{
  if (cur > lim)
    return sum;

  int rule = type ? cur % 2 == 0 : cur % 2 != 0;

  return rule ? sumUpto(lim, cur + 1, sum + cur, type) : sumUpto(lim, cur + 1, sum,
  ↪  type);
}

int main()
{
  int a;
  printf("Enter number to sum upto: ");
  scanf("%d", &a);
  printf("Sum of odd numbers upto %d: %d\n", a, sumUpto(a, 1, 0, 0));
  printf("Sum of even numbers upto %d: %d\n", a, sumUpto(a, 1, 0, 1));
  return 0;
}
```

Program Output :



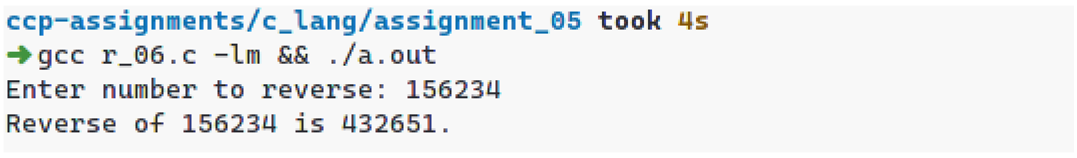
```
ccp-assignments/c_lang/assignment_05
➜gcc r_05.c -lm && ./a.out
Enter number to sum upto: 10
Sum of odd numbers upto 10: 25
Sum of even numbers upto 10: 30
```

2.6 Write a C program to find reverse of any number using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Reverse a number.
 * @param val current integer
 * @param rev current reversed integer
*/
int reverseInt(int val, int rev)
{
  if (val == 0)
    return rev;
  int rem = val % 10;
  return reverseInt(val / 10, (rev * 10) + rem);
}

int main()
{
  int a;
  printf("Enter number to reverse: ");
  scanf("%d", &a);
  printf("Reverse of %d is %d.\n", a, reverseInt(a, 0));
  return 0;
}
```

Program Output :



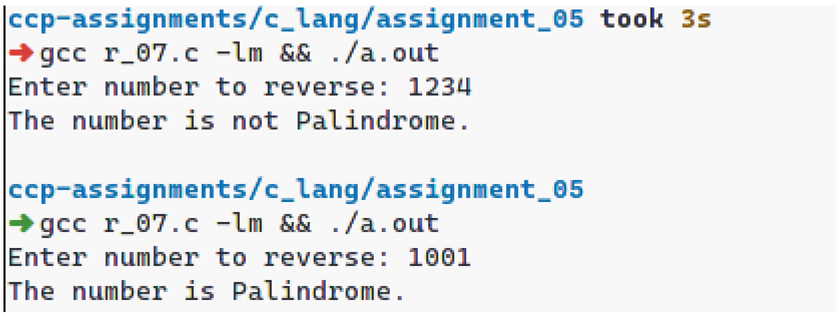
```
ccp-assignments/c_lang/assignment_05 took 4s
➜ gcc r_06.c -lm && ./a.out
Enter number to reverse: 156234
Reverse of 156234 is 432651.
```

2.7 Write a C program to check whether a number is palindrome or not using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Check if a number is a Palindrome.
 * @param val current integer
 * @param rev current reversed integer
 * @param init immutable initial integer
*/
int isPalindrome(int val, int rev, int init)
{
  if (val == 0)
    return rev == init ? 1 : 0;
  int rem = val % 10;
  return isPalindrome(val / 10, (rev * 10) + rem, init);
}

int main()
{
  int a;
  printf("Enter number to reverse: ");
  scanf("%d", &a);
  printf("The number %s Palindrome.\n", isPalindrome(a, 0, a) ? "is" : "is not");
  return 0;
}
```

Program Output :



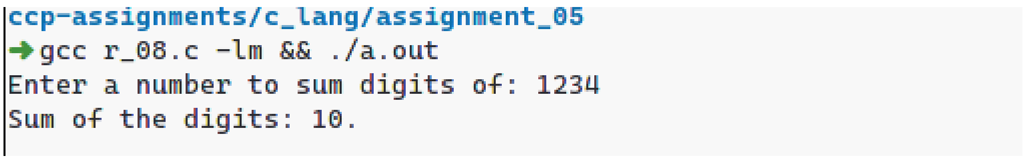
```
ccp-assignments/c_lang/assignment_05 took 3s
➜gcc r_07.c -lm && ./a.out
Enter number to reverse: 1234
The number is not Palindrome.

ccp-assignments/c_lang/assignment_05
➜gcc r_07.c -lm && ./a.out
Enter number to reverse: 1001
The number is Palindrome.
```

2.8 Write a C program to find sum of digits of a given number using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Sum of the digits of the given number.
 * @param val current integer
 * @param sum sum upto digits not in [val]
 */
int sumOfDigits(int val, int sum)
{
  if (val == 0)
    return sum;
  return sumOfDigits(val / 10, sum + (val % 10));
}

int main()
{
  int a;
  printf("Enter a number to sum digits of: ");
  scanf("%d", &a);
  printf("Sum of the digits: %d.\n", sumOfDigits(a, 0));
  return 0;
}
```

Program Output :



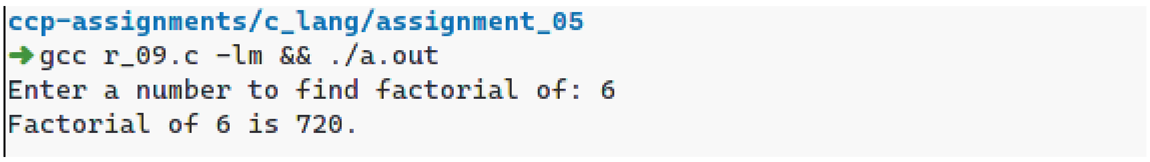
```
ccp-assignments/c_lang/assignment_05
➜gcc r_08.c -lm && ./a.out
Enter a number to sum digits of: 1234
Sum of the digits: 10.
```

2.9 Write a C program to find factorial of any number using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Find factorial of the given number.
 * @param cur current integer
 * @param val factorial upto digits not in [cur]
 */
int factorialOf(int cur, int val)
{
  if (cur <= 1)
    return val;
  return factorialOf(cur - 1, val * cur);
}

int main()
{
  int a;
  printf("Enter a number to find factorial of: ");
  scanf("%d", &a);
  printf("Factorial of %d is %d.\n", a, factorialOf(a, 1));
  return 0;
}
```

Program Output :



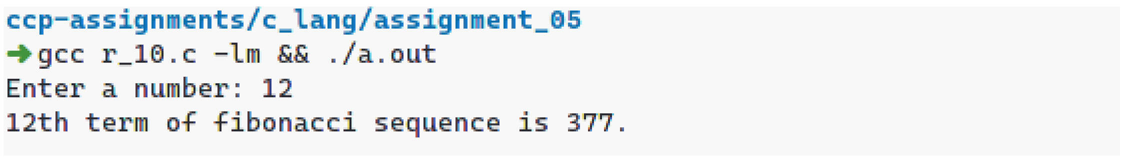
```
ccp-assignments/c_lang/assignment_05
➜gcc r_09.c -lm && ./a.out
Enter a number to find factorial of: 6
Factorial of 6 is 720.
```

2.10 Write a C program to generate nth Fibonacci term using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Find nth term in fibonacci sequence.
 * @param first First term of said sequence
 * @param Second second term of said sequence
 * @param n `n` of nth
 * @param c current index
*/
int getNthFib(int first, int second, int n, int c)
{
  if (n == c)
    return second;
  return getNthFib(second, first + second, n, c + 1);
}

int main()
{
  int a;
  printf("Enter a number: ");
  scanf("%d", &a);
  printf("%dth term of fibonacci sequence is %d.\n", a, getNthFib(1, 1, a, 0));
  return 0;
}
```

Program Output :



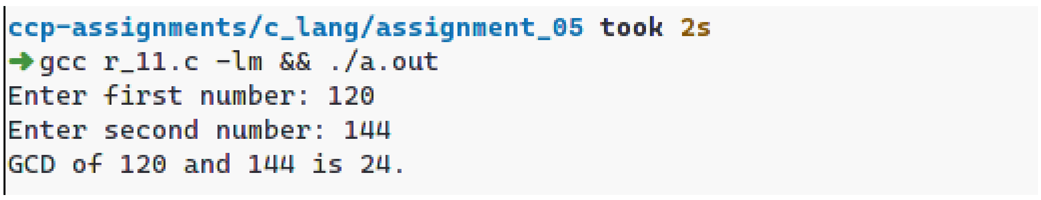
```
ccp-assignments/c_lang/assignment_05
➜gcc r_10.c -lm && ./a.out
Enter a number: 12
12th term of fibonacci sequence is 377.
```

2.11 Write a C program to find GCD (HCF) of two numbers using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Find the greatest common divisor of two numbers.
 * @param first first number
 * @param second second number
 * @param last last found common divisor
 * @param cur current iterator
*/
int findGCD(int first, int second, int last, int cur)
{
  int min = first < second ? first : second;
  if (cur > min)
    return last;
  int common = 0;
  if (first % cur == 0 && second % cur == 0)
    common = cur;
  return findGCD(first, second, common ? common : last, cur + 1);
}

int main()
{
  int a, b;
  printf("Enter first number: ");
  scanf("%d", &a);
  printf("Enter second number: ");
  scanf("%d", &b);
  printf("GCD of %d and %d is %d.\n", a, b, findGCD(a, b, 1, 1));
  return 0;
}
```

Program Output :



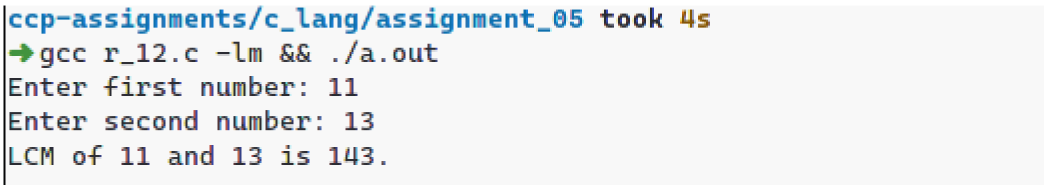
```
ccp-assignments/c_lang/assignment_05 took 2s
➜gcc r_11.c -lm && ./a.out
Enter first number: 120
Enter second number: 144
GCD of 120 and 144 is 24.
```

2.12 Write a C program to find LCM of two numbers using recursion.

Source Code :

```c
#include <stdio.h>

/**
 * Find the LCM of two numbers.
 * @param first first number
 * @param second second number
 * @param cur current iterator
*/
int findLCM(int first, int second, int cur)
{
  if (cur % first == 0 && cur % second == 0)
    return cur;
  return findLCM(first, second, cur + 1);
}

int main()
{
  int a, b;
  printf("Enter first number: ");
  scanf("%d", &a);
  printf("Enter second number: ");
  scanf("%d", &b);
  printf("LCM of %d and %d is %d.\n", a, b, findLCM(a, b, a > b ? a : b));
  return 0;
}
```

Program Output :



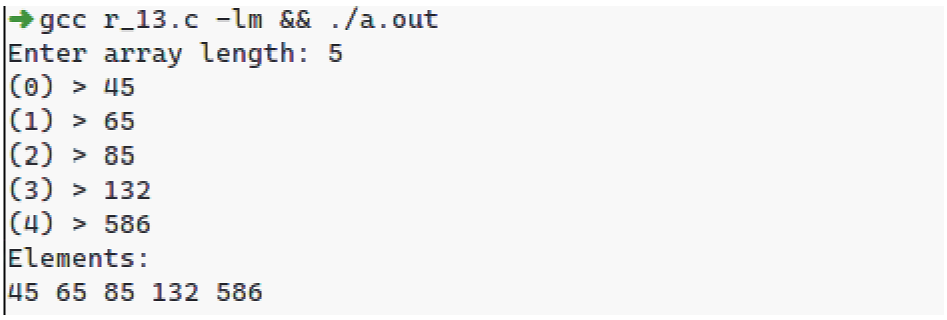
```
ccp-assignments/c_lang/assignment_05 took 4s
➜gcc r_12.c -lm && ./a.out
Enter first number: 11
Enter second number: 13
LCM of 11 and 13 is 143.
```

2.13 Write a C program to display all array elements using recursion.

Source Code :

```c
#include <stdio.h>

void printArray(int arr[], int length, int cur)
{
  if (cur >= length)
    return;
  printf("%d ", arr[cur]);
  return printArray(arr, length, cur + 1);
}

int main()
{
  int a[100], l;

  printf("Enter array length: ");
  scanf("%d", &l);

  for (int i = 0; i < l; i++)
  {
    printf("(%d) > ", i);
    scanf("%d", &a[i]);
  }

  printf("Elements: \n");
  printArray(a, l, 0);

  return 0;
}
```

Program Output :



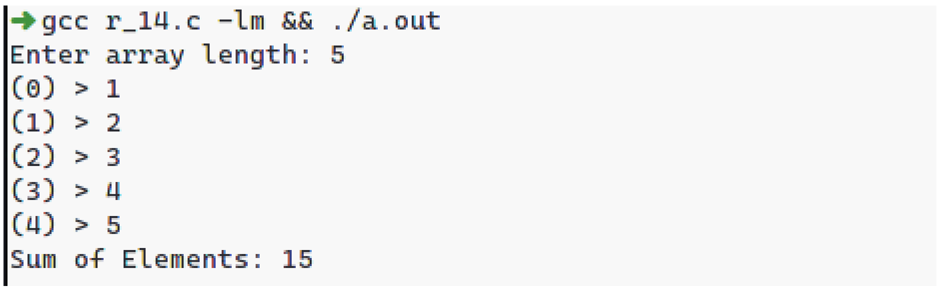
```
➜gcc r_13.c -lm && ./a.out
Enter array length: 5
(0) > 45
(1) > 65
(2) > 85
(3) > 132
(4) > 586
Elements:
45 65 85 132 586
```

2.14 Write a C program to find sum of elements of array using recursion.

Source Code :

```c
#include <stdio.h>

int sumOfElements(int arr[], int length, int cur, int sum)
{
  if (cur > length)
    return sum;
  return sumOfElements(arr, length, cur + 1, sum + cur);
}

int main()
{
  int a[100], l;

  printf("Enter array length: ");
  scanf("%d", &l);

  for (int i = 0; i < l; i++)
  {
    printf("(%d) > ", i);
    scanf("%d", &a[i]);
  }

  printf("Sum of Elements: %d\n", sumOfElements(a, l, 0, 0));

  return 0;
}
```

Program Output :



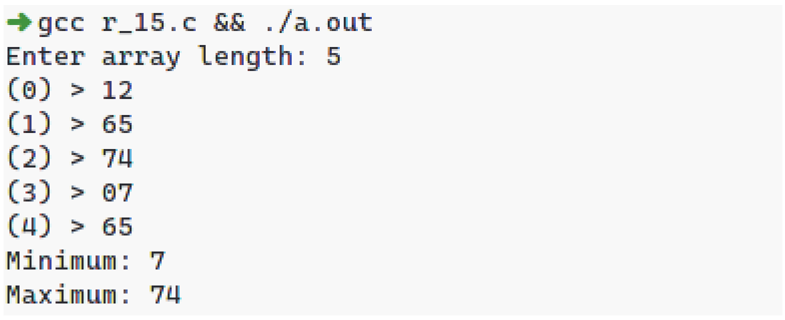
```
➜gcc r_14.c -lm && ./a.out
Enter array length: 5
(0) > 1
(1) > 2
(2) > 3
(3) > 4
(4) > 5
Sum of Elements: 15
```

2.15 Write a C program to find maximum and minimum elements in array using recursion.

Source Code :

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

// flag 0 = minimum, 1 = maximum
int minMax(int arr[], int length, int cur, int last, int flag)
{
  if (cur >= length)
    return last;
  int check = flag ? last < arr[cur] : last > arr[cur];
  return minMax(arr, length, cur + 1, check ? arr[cur] : last, flag);
}

int main()
{
  int a[100], l;

  printf("Enter array length: ");
  scanf("%d", &l);

  for (int i = 0; i < l; i++)
  {
    printf("(%d) > ", i);
    scanf("%d", &a[i]);
  }

  printf("Minimum: %d\n", minMax(a, l, 0, INT_MAX, 0));
  printf("Maximum: %d\n", minMax(a, l, 0, INT_MIN, 1));

  return 0;
}
```

Program Output :



```
➜gcc r_15.c && ./a.out
Enter array length: 5
(0) > 12
(1) > 65
(2) > 74
(3) > 07
(4) > 65
Minimum: 7
Maximum: 74
```