|  |  |
| --- | --- |
| **Name** | Mahadev Balla |
| **UID no.** | 2023300010 |
| **Experiment No.** | 3 |

|  |  |
| --- | --- |
| **AIM:** | Apply the concept of functions to incorporate modularity. |
| **Program 1** | |
| **PROBLEM STATEMENT :** | Write a modular code for checking whether the pair is amicable numbers or not. |
| **ALGORITHM:** | 1) DEFINE A FUNCTION ‘SUMFACTOR’ HAVING PARAMETER ‘INT X’ WHICH RETURNS THE VALUE OF SUM OF FACTORS OF X USING FOR LOOP AND PRINTS “SUM IS : val(sum)”  2) INITIALIZE “n1, n2, ans1, ans2”  3) INPUT “n1, n2”  4) CALL THE FUNCTION ‘SUMFACTOR’ PASSING ARGUMENT n1 AND ASSIGN THE VALUE val(sum) TO ans1  5) CALL THE FUNCTION ‘SUMFACTOR’ PASSING ARGUMENT n2 AND ASSIGN THE VALUE val(sum) TO ans2  6) IF(ans1=ans2) PRINT “n1 and n2 are amicable numbers”  ELSE PRINT “n1 and n2 are not amicable numbers” |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include <stdio.h>  int sumfactor(int n);  int main()  {  int n1, n2;  printf("Enter two numbers to check if they are amicable or not :");  scanf("%d %d",&n1,&n2);  int ans1 = sumfactor(n1);  int ans2 = sumfactor(n2);  if(ans1==ans2)  printf("%d and %d are amicable numbers",n1,n2);  else  printf("%d and %d are not amicable numbers",n1,n2);  return 0;  }  int sumfactor(int x)  {  int sum=0;  for(int i=1;i<=x;i++)  { if ((x%i) == 0)  { sum+=i;}  }  printf("Sum is : %d\n",sum);  return(sum);  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | Write a modular code to take range of numbers as input and print all the prime numbers within the range. |
| **ALGORITHM:** | 1) DEFINE A FUNCTION ‘PRINT\_PRIMES’ HAVING PARAMETERS ‘INT S,INT E’ WHICH PRINTS ALL THE PRIME NUMBERS WITHIN A GIVEN RANGE OF VALUES USING NESTED FOR LOOP  2) INITIALIZE “s, e”  3) INPUT “s, e”  4) CALL FUNCTION ‘PRINT\_PRIMES’ PASSING (s, e) AS THE ARGUMENTS |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include<stdio.h>  #include<math.h>  void print\_primes(int s,int e);  int main()  {  int s,e;  printf("Enter the range of values : ");  scanf("%d %d",&s,&e);  print\_primes(s,e);  return 0;  }  void print\_primes(int start,int end)  {  int cnt=0;  for (int i=start;i<=end;i++)  {  int j;  for (j=2;j<=sqrt(i);j++)  if (i%j==00)  {  break;  }  if (j>sqrt(i)&&i!=1)  printf("%d, ",i);  cnt++;  if (cnt==10)  {  printf("\n");  cnt=0;  }  }  } |
| **RESULT:** | |
| **Program 3** | |
| **PROBLEM STATEMENT:** | Write a modular code to swap two numbers using call by reference. |
| **ALGORITHM:** | 1) DEFINE A FUNCTION ‘SWAP’ HAVING PARAMETERS  (int \*x, int \*y) WHICH SWAPS TWO NUMBERS USING CALL BY REFERENCE  2) INITIALIZE “n1, n2”  3) INPUT “n1, n2”  4) PRINT “First number and second number before swapping : n1 and n2”  5) CALL FUNCTION ‘SWAP’ PASSING (n1, n2) AS THE ARGUMENTS  6) PRINT “First number and second number after swapping : n1 and n2” |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include <stdio.h>  void swap(int \*x, int \*y)  {  int t;  t=\*x;  \*x=\*y;  \*y=t;  }  int main ()  {  int n1,n2;  printf("Enter two numbers : ");  scanf("%d %d",&n1,&n2);  printf("First number and second number before swapping : %d and %d\n",n1,n2);  swap(&n1,&n2);  printf("First number and second number after swapping : %d and %d\n",n1,n2);  return 0;  } |
| **RESULT:** | |
| **CONCLUSION:** | Studied the application of the concept of functions to incorporate modularity. |