|  |  |  |
| --- | --- | --- |
| **Name:**  **P K Navin Shrinivas** | **SRN:PES2UG20CS237** | **Section:G(B-2)** |
| **Date:20/05/21** | **Week Number:3** |

|  |  |
| --- | --- |
| **1** | Write a function to reverse a given number and check whether a given number is palindrome or not.Input:Enter the number121Output:The Number 121 is PalindromeInput:Enter the numberOutput:123Number 123 is Not Palindrome |
|  | **Program:**  #include<stdio.h>  #include<stdbool.h>  #include<string.h>  bool reversematch(char *a*[]) {  char copy[100000];  for (int i = 0; i < strlen(a); i++) {  copy[i] = a[strlen(a) - i - 1];  }  for (int i = 0; i < strlen(a); i++) {  if (a[i] != copy[i]) {  return false;  }  }  return true;  }  int main() {  char c[100000];  printf("Enter a number :");  scanf("*%*[^\n]%c", c);  if (reversematch(c))  printf("Yes %s is a palindrome!", c);  else  printf("No,%s is not a palindrome!", c);  return 0;  } |
|  | **Output Screenshot:**  **1** |
| **2** | Write a C program to compute GCD of three numbers using functions. **Input:**  Enter the values of a,b and c  10 4 16  **Output:**  GCD(10,4,16)=2 |
|  | **Program:**  #include<stdio.h>  #include<stdbool.h>  int recursegdc(int *a*, int *b*) {  if (a == 0)  return b;  recursegdc(b % a, a);  }  int threegcd(int *a*[]) {  int init = a[0];  for (int i = 0; i < 3; i++) {  init = recursegdc(a[i], init);  if (init == 1)  return 1;  }  return init;  }  int main() {  int a[3];  printf("Enter 3 values:");  for (int i = 0; i < 3; i++)  scanf("%i", a + i);  printf("GCD(%i,%i,%i)=%i", a[0], a[1], a[2], threegcd(a));  } |
|  | **Output Screenshot:**  **2** |
| **3** | Write a program in C to check Armstrong and perfect numbers using functions.  **Input:**  Input any number: 153  **Output:**  The 153 is an Armstrong number.  The 153 is not a Perfect number.  **Input:**  Input any number: 28  **Output:**  The 28 is not an Armstrong number.  The 28 is a Perfect number. |
|  | **Program:**  #include<stdio.h>  #include<stdbool.h>  void recurseperfect(int \* *copy*, int *n*, int \* *sum*) {  if (n == 1) {  \* sum + 1;  return;  }  if ( \* copy % n == 0)  \*sum = \* sum + n;  recurseperfect(copy, n - 1, sum);  }  bool armstrong(int *n*) {  int sum = 0;  int copy = n;  while (n != 0) {  sum += (n % 10) \* (n % 10) \* (n % 10);  n /= 10;  }  if (sum == copy)  return true;  else {  return false;  }  }  int main() {  int n;  int sum = 0;  printf("Enter a number :");  scanf("%i", & n);  recurseperfect( & n, n - 1, & sum);  sum++;  if (armstrong(n))  printf("%i is is a armstrong number!\n", n);  else  printf("%i is is a not a armstrong number!\n", n);  if (sum == n)  printf("%i is is a perfect number!", sum);  else  printf("%i is not a perfect number!", n);  } |
|  | **Output Screenshot:**  **3** |
| **4** | Write a program in C to check whether a number is a prime number or not using function  **Input:**  Input a positive number : 12  **Output:**  The number 12 is not a prime number  **Input:**  Input a positive number : 13  **Output:**  The number 13 is a prime number |
|  | **Program:**  //if a number is not a prime atleast one of its divisor is present which is less than the root(number)  #include<stdio.h>  #include<stdbool.h>  #include<math.h>  bool checkprime(int *n*) {  for (int i = sqrt(n) + 1; i > 1; i--) {  if (n % i == 0)  return false;  }  return true;  }  int main() {  int n;  printf("Enter a number :");  scanf("%i", & n);  if (checkprime(n))  printf("The number %i is a prime", n);  else  printf("The number %i is not a prime", n);  } |
|  | **Output Screenshot:**  **4** |
| **5** | Write a program in C to convert decimal number to octal number using function  **Input:**  Input any decimal number : 25  **Output:**  Equivalent Octal Number: 31  **Input:**  Input any decimal number : 15  **Output:**  Equivalent Octal Number: 17 |
|  | **Program:**  #include<stdio.h>  int c = 0;  void recurseoctal(int \* *n*, int *rems*[]) {  if ( \* n < 8) {  rems[c] = \* n;  c++;  } else {  rems[c] = \* n % 8;  c++;  \* n /= 8;  recurseoctal(n, rems);  }  }  int main() {  int n;  printf("Enter a number[BASE 10] :");  scanf("%i", & n);  int copy = n;  int rems[1000000];  recurseoctal( & n, rems);  printf("%i in octal [BASE 8] :", copy);  for (int i = c - 1; i >= 0; i--)  printf("%i", rems[i]);  } |
|  | **Output Screenshot:**  **5** |
| **6** | Write a program in C to find the sum of the series 1!/1+2!/2+3!/3+4!/4+5!/5 using function.  **Output:**  The sum of the series is : 34 |
|  | **Program:**  #include<stdio.h>  int recursefactorial(int *n*) {  if (n == 1)  return 1;  else  return n \* recursefactorial(n - 1);  }  int main() {  int sum = 0;  for (int i = 1; i <= 5; i++) {  sum += recursefactorial(i) / i;  }  printf("sum of the requrired series is :%i", sum);  } |
|  | **Output Screenshot:**  **6** |
| **1** | **Practice Programs**  Write a program to display Fibonacci series in C within a range using a function  **Input:**  Enter range: 5  **Output:**  The fibonacci series is:  0 1 1 2 3 5 |
|  | **Program:**  #include<stdio.h>  int main() {  int first = 0;  int second = 1;  int n;  printf("Enter range for fibbonaci :");  scanf("%i", & n);  printf("%i ", first);  while (second <= n) {  printf("%i ", second);  int temp = first;  first = second;  second = temp + second;  }  } |
|  | **Output Screenshot:**  **7** |
| 2 | Write a program to check triangle validity when angles are given using functions.  **Input:**  Enter three angles of triangle:  30  40  60  **Output:**  Triangle is not valid  **Input:**  Enter three angles of triangle:  30  60  90  **Output:**  Triangle is valid |
|  | **Program:**  #include<stdio.h>  int main() {  int a, b, c;  printf("Enter the three angles of the triangle :");  scanf("%i %i %i", & a, & b, & c);  if (a + b + c == 180)  printf("Triagle is valid!");  else  printf("Triangle is not valid");  } |
|  | **Output Screenshot:**  **8** |