```
int fact(int n);


int main() {
   int num,f;
   printf("Enter a number: ");
   scanf("%d", &num);
   f=fact(num);
   printf("Factorial of %d=%d\n",num,f);
   return 0;
}
int fact(int n) {
   if(n)
     return n*fact(n-1);
   else
     return 1;
}


//with pointers
#include <stdio.h>


int fact(int *n);


int main() {
   int num,f;
   printf("Enter a number: ");
   scanf("%d", &num);
   f=fact(&num);
   printf("Factorial =%d\n",f);
   return 0;
}
int fact(int *n) {
   if(*n>0){
       int f1=*n;
       (*n)--;
       return f1 *fact(n);
   }
   else
     return 1;
}


#include <stdio.h>

int fibonacci(int n);

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

   if (n < 0) {
       printf("Fibonacci is not defined for negative integers.\n");
   } else {
       int result = fibonacci(n);
       printf("Fibonacci number %d is %d\n", n, result);
   }

   return 0;
}


int fibonacci(int n) {
   if (n == 0) {
       return 0;
   } else if (n == 1) {
       return 1;
   } else {
       // Recursive case: F(n) = F(n-1) + F(n-2)
       return fibonacci(n - 1) + fibonacci(n - 2);
   }
}

//with pointers
#include <stdio.h>
int fibonacci(int *n);

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

   if (num < 0) {
       printf("Fibonacci is not defined for negative integers.\n");
   } else {
       int result = fibonacci(&num);   // Pass the address of num to the recursive function
       printf("Fibonacci number %d is %d\n", num, result);
   }

   return 0;
}

int fibonacci(int *n) {
   if (*n == 0) {
       return 0;
   } else if (*n == 1) {
       return 1;
   } else {
       int n_minus_1 = *n - 1;  // Calculate n-1
       int n_minus_2 = *n - 2;  // Calculate n-2
       return fibonacci(&n_minus_1) + fibonacci(&n_minus_2);
   }
}
#include <stdio.h>

int sumOfDigits(int n);

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

   if (num < 0) {
       printf(" enter a positive integer.\n");
   } else {
       int result = sumOfDigits(num);
       printf("Sum of digits of %d is %d\n", num, result);
   }

   return 0;
}

int sumOfDigits(int n) {
   if (n == 0) {
       return 0;
   } else {
       return (n % 10) + sumOfDigits(n / 10); // Adding the last digit to the sum of remaining digits
   }
}

// with pointers
#include <stdio.h>

int sumOfDigits(int *n);

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

   if (num < 0) {
       printf("enter a positive integer.\n");
   } else {
       result = sumOfDigits(&num);
       printf("Sum of digits = %d\n", result);
   }

   return 0;
}
int sumOfDigits(int *n) {
   if (*n == 0) {
       return 0;
   } else {
       int lastDigit = *n % 10;
       *n = *n / 10;
       return lastDigit + sumOfDigits(n);
   }
}

#include <stdio.h>
#include <string.h>

void reverseString(char *str, int start, int end);

int main() {
   char str[100];
   printf("Enter a string: ");
   scanf("%s", str);  // Read the string

   int len = strlen(str);
   reverseString(str, 0, len - 1);  // Call recursive function to reverse the string

   printf("Reversed string: %s\n", str);

   return 0;
}
void reverseString(char *str, int start, int end) {
   if (start >= end) {
       return;
   }
   // Swap characters at start and end
   char temp = str[start];
   str[start] = str[end];
   str[end] = temp;

   // Recursive call to reverse the inner substring
   reverseString(str, start + 1, end - 1);
```

```
}

//with pointers

#include <stdio.h>
#include <string.h>
void reverseString(char  *str, char *start, char *end);

int main() {
    char str[100];
    printf("Enter  a string:  ");
    scanf("%s",  str);
    reverseString(str,  str, str + strlen(str)  - 1);

    printf("Reversed  string:  %s\n", str);

    return 0;
}
void reverseString(char  *str, char *start, char *end) {
    if (start >= end) {
        return;
    }

    // Swap characters at start and end
    char temp = *start;
    *start = *end;
    *end = temp;

    // Recursive  call with the next start and previous  end pointers
    reverseString(str,  start + 1, end - 1);
}
```

```
#include <stdio.h>
int power(int x, int n);


int main() {
   int x, n;
   printf("Enter base number: ");
   scanf("%d", &x);
   printf("Enter exponent: ");
   scanf("%d", &n);


   int result = power(x, n);
   printf("%d raised to the power %d is: %d\n", x, n, result);


   return 0;
}
int power(int x, int n) {
   if (n == 0) {
      return 1;
   } else {
      return x * power(x, n - 1);  // Recursive case: x^n = x * x^(n-1)
   }
}


//with pointers
#include <stdio.h>
int power(int *x, int *n);


int main() {
   int x, n;
   printf("Enter base number: ");
```

```
    scanf("%d", &x);
    printf("Enter exponent: ");
    scanf("%d", &n);

    int result = power(&x, &n);
    printf("%d raised to the power %d is: %d\n", x, n, result);

    return 0;
}
int power(int *x, int *n) {
    if (*n == 0) {
        return 1;
    } else {
        int temp = *n - 1;  // Decrement the exponent
        return *x * power(x, &temp);  // Recursive case: x^n = x * x^(n-1)
    }
}

#include <stdio.h>
int gcd(int x, int y);

int main() {
    int x, y;
    printf("Enter two numbers: ");
    scanf("%d %d", &x, &y);

    int result = gcd(x, y);
    printf("The GCD of %d and %d is: %d\n", x, y, result);

    return 0;
}
```

```
int gcd(int x, int y) {
    if (y == 0) {
        return x;
    } else {
        return gcd(y, x % y);
    }
}

//with pointers
#include <stdio.h>
int gcd(int *x, int *y);

int main() {
    int x, y;
    printf("Enter two numbers: ");
    scanf("%d %d", &x, &y);

    int result = gcd(&x, &y);
    printf("The GCD of %d and %d is: %d\n", x, y, result);

    return 0;
}
int gcd(int *x, int *y) {
    if (*y == 0) {
        return *x;
    } else {
        int remainder = *x % *y;  // Calculate the remainder
        return gcd(y, &remainder);   // Recursive case: GCD(y, remainder)
    }
}
```

```
#include <stdio.h>
int countOccurrences(char  *str, char target);


int main() {
    char str[100], target;
    printf("Enter  a string:  ");
    scanf("%s", str);
    printf("Enter  a character to count:  ");
    scanf(" %c", &target);


    int result = countOccurrences(str,  target);
    printf("The  character '%c' appears %d times in the string.\n", target, result);


    return 0;
}
int countOccurrences(char  *str, char target) {
    if (*str == 0) {
        return 0;
    } else {
        if (*str == target) {
            return 1 + countOccurrences(str  + 1, target);
        } else {
            return countOccurrences(str  + 1, target);
        }
    }
}


//with pointers
#include <stdio.h>
int countOccurrences(char  *str, char target);
int main() {
    char str[100], target;
    printf("Enter a string: ");
    scanf("%s", str);
    printf("Enter a character to count: ");
    scanf(" %c", &target);

    int result = countOccurrences(str, target);
    printf("The character '%c' appears %d times in the string.\n", target, result);

    return 0;
}
int countOccurrences(char *str, char target) {
    if (*str == '\0') { // Base case: end of string
        return 0;
    } else {
        if (*str == target) {
            return 1 + countOccurrences(str + 1, target);
        } else {
            return countOccurrences(str + 1, target);
        }
    }
}

#include <stdio.h>
#include <string.h>
int isPalindrome(char str[], int start, int end);

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);

    int length = strlen(str);
    if (isPalindrome(str, 0, length - 1)) {
        printf("The string is a palindrome.\n");
    } else {
        printf("The string is not a palindrome.\n");
    }

    return 0;
}
int isPalindrome(char str[], int start, int end) {
    if (start >= end) {
        return 1;
    }
    if (str[start] != str[end]) {
        return 0;  // It's not a palindrome
    }
    return isPalindrome(str, start + 1, end - 1);
}

// with pointers
#include <stdio.h>

int isPalindrome(char *str, char *start, char *end);

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    char *start = str;
    char *end = str;

    while (*end)
        end++;

    end--;
    if (isPalindrome(str, start, end))
        printf("The string is a palindrome.\n");
    else
        printf("The string is not a palindrome.\n");

    return 0;
}

int isPalindrome(char *str, char *start, char *end) {
    if (start >= end)
        return 1;

    if (*start != *end)
        return 0;

    return isPalindrome(str, start + 1, end - 1);
}

#include <stdio.h>

int stringLength(char str[]);

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);

    int length = stringLength(str);
    printf("Length of the string is: %d\n", length);

    return 0;
}

int stringLength(char str[]) {
    if (str[0] == '\0')
        return 0;
    return 1 + stringLength(str + 1);
}

//with pointers
#include <stdio.h>

int stringLength(char *str);

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);

    int length = stringLength(str);
    printf("Length of the string is: %d\n", length);

    return 0;
}
```

```
int stringLength(char  *str) {
   if (*str == '\0')
      return 0;
   return 1 + stringLength(str  + 1);
}



#include <stdio.h>


int isPrime(int  num, int divisor);


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


   if (isPrime(num,  num / 2))
      printf("%d  is a prime  number.\n",  num);
   else
      printf("%d  is not  a prime  number.\n");


   return 0;
}


int isPrime(int  num, int divisor) {
   if (divisor  == 1)
      return 1;


   if (num % divisor  == 0)
      return 0;
```

```
    return isPrime(num, divisor - 1);
}

//with pointers
#include <stdio.h>

int isPrime(int *num, int divisor);

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

    if (isPrime(&num, num / 2))
        printf("%d is a prime number.\n", num);
    else
        printf("%d is not a prime number.\n");

    return 0;
}

int isPrime(int *num, int divisor) {
    if (divisor == 1)
        return 1;

    if (*num % divisor == 0)
        return 0;

    return isPrime(num, divisor - 1);
}
```

```
#include <stdio.h>

void printReverse(int num);

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

    printf("Numbers in reverse order: ");
    printReverse(num);
    printf("\n");

    return 0;
}

void printReverse(int num) {
    if (num == 0)
        return;

    printf("%d ", num);
    printReverse(num - 1);
}

//with pointers
#include <stdio.h>

void printReverse(int *num);

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

    printf("Numbers in reverse order: ");
    printReverse(&num);
    printf("\n");

    return 0;
}

void printReverse(int *num) {
    if (*num == 0)
        return;

    printf("%d ", *num);
    (*num)--;
    printReverse(num);
}

#include <stdio.h>

int arraySum(int arr[], int size);

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    int sum = arraySum(arr, size);
    printf("Sum of array elements: %d\n", sum);
    return 0;
}

int arraySum(int arr[], int size) {
    if (size == 0)
        return 0;

    return arr[size - 1] + arraySum(arr, size - 1);
}
```

```
//with pointers

#include <stdio.h>

int arraySum(int *arr, int size);

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    int sum = arraySum(arr, size);
    printf("Sum of array elements: %d\n", sum);

    return 0;
}

int arraySum(int *arr, int size) {
    if (size == 0)
        return 0;

    return *(arr + size - 1) + arraySum(arr, size - 1);
```

```
}



#include <stdio.h>
#include <string.h>

void permute(char str[], int left, int right);

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);

    printf("Permutations of the string are:\n");
    permute(str, 0, strlen(str) - 1);

    return 0;
}

void permute(char str[], int left, int right) {
    if (left == right) {
        printf("%s\n", str);
    } else {
        for (int i = left; i <= right; i++) {
            char temp = str[left];
            str[left] = str[i];
            str[i] = temp;

            permute(str, left + 1, right);

            temp = str[left];
            str[left] = str[i];
            str[i] = temp;
        }
    }
}

//with pointers

#include <stdio.h>

void permute(char *str, int left, int right);

int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);

    printf("Permutations of the string are:\n");
    permute(str, 0, strlen(str) - 1);

    return 0;
}

void permute(char *str, int left, int right) {
    if (left == right) {
        printf("%s\n", str);
    } else {
        for (int i = left; i <= right; i++) {
            char temp = *(str + left);
            *(str + left) = *(str + i);
            *(str + i) = temp;
            permute(str, left + 1, right);

            temp = *(str + left);
            *(str + left) = *(str + i);
            *(str + i) = temp;
        }
    }
}
```

20->14>21->45->89->56->63->72

1.diplay the linked list

2. count the number of elements present in the link list na dprint it

3. summ up of all the lements in the linked list

4. FInd the maximum element

5, find the minmum element in the linked list

6. Search for a particullar element whether it is present in the linked list.

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

struct Node {
    int data;
    struct Node *next;
};
```

```
void Display(struct Node *);
int CountNodes(struct Node *);
int SumElements(struct Node *);
int FindMax(struct Node *);
int FindMin(struct Node *);
int SearchElement(struct Node *, int);

int main() {
    struct Node *first = NULL;

    first = (struct Node *)malloc(sizeof(struct Node));
    first->data = 20;
    first->next = NULL;

    struct Node *second = (struct Node *)malloc(sizeof(struct Node));
    second->data = 14;
    second->next = NULL;
    first->next = second;

    struct Node *third = (struct Node *)malloc(sizeof(struct Node));
    third->data = 21;
    third->next = NULL;
    second->next = third;

    struct Node *fourth = (struct Node *)malloc(sizeof(struct Node));
    fourth->data = 45;
    fourth->next = NULL;
    third->next = fourth;

    struct Node *fifth = (struct Node *)malloc(sizeof(struct Node));
```

```
fifth->data = 89;

fifth->next = NULL;

fourth->next = fifth;


struct Node *sixth = (struct Node *)malloc(sizeof(struct Node));

sixth->data = 56;

sixth->next = NULL;

fifth->next = sixth;


struct Node *seventh = (struct Node *)malloc(sizeof(struct Node));

seventh->data = 63;

seventh->next = NULL;

sixth->next = seventh;


struct Node *eighth = (struct Node *)malloc(sizeof(struct Node));

eighth->data = 72;

eighth->next = NULL;

seventh->next = eighth;


Display(first);


int count = CountNodes(first);

printf("Number of elements in the linked list: %d\n", count);


int sum = SumElements(first);

printf("Sum of all elements in the linked list: %d\n", sum);


int max = FindMax(first);

printf("Maximum element in the linked list: %d\n", max);


int min = FindMin(first);
```

```
    printf("Minimum  element in the linked list: %d\n", min);


    int element;
    printf("Enter the element to search: ");
    scanf("%d", &element);
    if (SearchElement(first, element)) {
        printf("Element %d is present in the linked list.\n", element);
    } else {
        printf("Element %d is not present in the linked list.\n", element);
    }


    return 0;
}


void Display(struct Node *p) {
    while (p != NULL) {
        printf("%d -> ", p->data);
        p = p->next;
    }
    printf("\n");
}


int CountNodes(struct Node *p) {
    int count = 0;
    while (p != NULL) {
        count++;
        p = p->next;
    }
    return count;
}
```

```
int SumElements(struct  Node *p) {
    int sum = 0;
    while (p != NULL) {
        sum += p->data;
        p = p->next;
    }
    return sum;
}

int FindMax(struct  Node *p) {
    int max = p->data;
    while (p != NULL) {
        if (p->data  > max) {
            max = p->data;
        }
        p = p->next;
    }
    return max;
}

int FindMin(struct  Node *p) {
    int min = p->data;
    while (p != NULL) {
        if (p->data  < min) {
            min = p->data;
        }
        p = p->next;
    }
    return min;
}
```

```
int SearchElement(struct Node *p, int element) {
   while (p != NULL) {
      if (p->data == element) {
         return 1; // Element found
      }
      p = p->next;
   }
   return 0; // Element not found
}
```