**Module-1**

1. **Sum and difference of two numbers**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

int a,b;

float c,d;

scanf("%d %d",&a,&b);

scanf("%f %f",&c ,&d);

int int\_sum = a+b;

int int\_diff = a-b;

float float\_sum = c+d;

float float\_diff = c-d;

printf("%d %d\n",int\_sum,int\_diff);

printf("%0.1f %0.1f", float\_sum,float\_diff);

return 0;

}

A black screen with white text

Description automatically generated

**2. Playing with characters**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

char ch;

char s[24];

char t[100];

scanf("%c", &ch);

scanf("%s", s);

getchar();

scanf("%[^\n]%\*c", t);

printf("%c\n", ch);

printf("%s\n", s);

printf("%s\n", t);

return 0;

}

A screenshot of a computer

Description automatically generated

**3.Conditional statements in C**

#include <assert.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

int main()

{

char\* n\_endptr;

char\* n\_str = readline();

int n = strtol(n\_str, &n\_endptr, 10);

if (n\_endptr == n\_str || \*n\_endptr != '\0') { exit(EXIT\_FAILURE); }

if (n==1){

printf("one");

}

else if(n==2){

printf("two");

}

else if(n==3){

printf("three");

}

else if(n==4){

printf("four");

}

else if(n==5){

printf("five");

}

else if(n==6){

printf("six");

}

else if(n==7){

printf("seven");

}

else if(n==8){

printf("eight");

}

else if(n==9){

printf("nine");

}

else if(n>9){

printf("Greater than 9");

}

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char data[1024];

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) { break; }

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') { break; }

size\_t new\_length = alloc\_length << 1;

if (!data) { break; }

alloc\_length = new\_length;

}

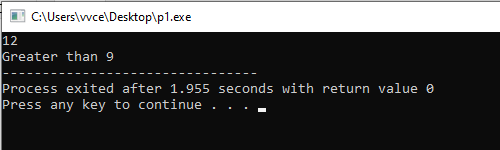
if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

}

return data;

}



**4. Valid Paranthesis**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SIZE 100

// Global variables for stack and top

char stack[MAX\_SIZE];

int top = -1;

// Function to push a character onto the stack

void push(char data) {

if (top == MAX\_SIZE - 1) {

printf("Overflow stack!\n");

return;

}

top++;

stack[top] = data;

}

// Function to pop a character from the stack

char pop() {

if (top == -1) {

printf("Empty stack!\n");

return ' ';

}

char data = stack[top];

top--;

return data;

}

// Function to check if two characters form a matching pair of parentheses

int is\_matching\_pair(char char1, char char2) {

if (char1 == '(' && char2 == ')') {

return 1;

} else if (char1 == '[' && char2 == ']') {

return 1;

} else if (char1 == '{' && char2 == '}') {

return 1;

} else {

return 0;

}

}

// Function to check if the expression is balanced

int isBalanced(char\* text) {

int i;

for (i = 0; i < strlen(text); i++) {

if (text[i] == '(' || text[i] == '[' || text[i] == '{') {

push(text[i]);

} else if (text[i] == ')' || text[i] == ']' || text[i] == '}') {

if (top == -1) {

return 0; // If no opening bracket is present

} else if (!is\_matching\_pair(pop(), text[i])) {

return 0; // If closing bracket doesn't match the last opening bracket

}

}

}

if (top == -1) {

return 1; // If the stack is empty, the expression is balanced

} else {

return 0; // If the stack is not empty, the expression is not balanced

}

}

// Main function

int main() {

char text[MAX\_SIZE];

printf("Input an expression in parentheses: ");

scanf("%s", text);

// Check if the expression is balanced or not

if (isBalanced(text)) {

printf("The expression is balanced.\n");

} else {

printf("The expression is not balanced.\n");

}

return 0;

}

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**5. Bitwise Operators**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

void calculate\_the\_maximum(int n, int k) {

   int max\_and = 0, max\_or = 0, max\_xor = 0;

    for (int i = 1; i <= n; i++) {

        for (int j = i + 1; j <= n; j++) {

            int temp\_and = i & j;

            int temp\_or = i | j;

            int temp\_xor = i ^ j;

            if (temp\_and > max\_and && temp\_and < k) {

                max\_and = temp\_and;

            }

            if (temp\_or > max\_or && temp\_or < k) {

                max\_or = temp\_or;

            }

            if (temp\_xor > max\_xor && temp\_xor < k) {

                max\_xor = temp\_xor;

            }

        }

    }

    printf("%d\n%d\n%d", max\_and, max\_or, max\_xor);

}

int main() {

    int n, k;

    scanf("%d %d", &n, &k);

    calculate\_the\_maximum(n, k);

    return 0;

}

**A black screen with white x and white crosses

Description automatically generated**

**MODULE -2**

**1. Printing Patterns Using Loops**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

     int n;

    scanf("%d", &n);

int len = 2\*n - 1;

    for (int i = 0; i < len; i++) {

        for (int j = 0; j < len; j++) {

            int min = i < j ? i : j;

            min = min < len-i ? min : len-i-1;

            min = min < len-j-1 ? min : len-j-1;

            printf("%d ", n-min);

        }

        printf("\n");

    }

      return 0;

**}**

**Output**

**A screenshot of a computer

Description automatically generated**

**2. Correctness and Loop Variant**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include <assert.h>

void insertionSort(int N, int arr[]) {

int i,j;

int value;

for(i=1;i<N;i++)

{

value=arr[i];

j=i-1;

while(j>=0 && value<arr[j])

{

arr[j+1]=arr[j];

j=j-1;

}

arr[j+1]=value;

}

for(j=0;j<N;j++)

{

printf("%d",arr[j]);

printf(" ");

}

}

int main(void) {

int N;

scanf("%d", &N);

int arr[N], i;

for(i = 0; i < N; i++) {

scanf("%d", &arr[i]);

}

insertionSort(N, arr);

return 0;

}

**A black screen with white text

Description automatically generated**

**3. Small And Large Triangle**

#include <stdio.h>

// Function to calculate the area of a triangle

float calculateArea(float base, float height) {

return (0.5 \* base \* height);

}

int main() {

float base1, height1, base2, height2, area1, area2;

// Input for the first triangle

printf("Enter the base and height of the first triangle: ");

scanf("%f %f", &base1, &height1);

// Input for the second triangle

printf("Enter the base and height of the second triangle: ");

scanf("%f %f", &base2, &height2);

// Calculate the areas of both triangles

area1 = calculateArea(base1, height1);

area2 = calculateArea(base2, height2);

// Compare the areas and determine which triangle is larger

printf("\nArea of first triangle: %.2f", area1);

printf("\nArea of second triangle: %.2f", area2);

if (area1 > area2) {

printf("\nThe first triangle is larger.\n");

printf("The second triangle is smaller.\n");

} else if (area1 < area2) {

printf("\nThe second triangle is larger.\n");

printf("The first triangle is smaller.\n");

} else {

printf("\nBoth triangles have the same area.\n");

}

return 0;

}

**A screenshot of a computer

Description automatically generated**

**4. Happy Numbers**

#include<stdio.h>

int main()

{

int num,sum=0;

printf("Enter the number");

scanf("%d",&num);

while((num!=1)&& (num!=4)){

while(num>0){

sum=sum+((num%10)\*(num%10));

num=num/10;

}

num=sum;

sum=0;

}

if(num==1){

printf("Happy Number");

}

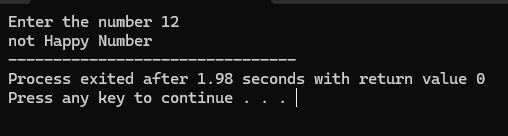
else{

printf("not Happy Number");

}

return 0;

}



**5. Triangle Numbers**

#include <stdio.h>

// Function to calculate the nth triangular number

int triangularNumber(int n) {

return (n \* (n + 1)) / 2;

}

int main() {

int n, i;

// Input for how many triangular numbers to generate

printf("Enter the number of triangular numbers to generate: ");

scanf("%d", &n);

// Generate and display the first n triangular numbers

printf("The first %d triangular numbers are:\n", n);

for (i = 1; i <= n; i++) {

printf("%d ", triangularNumber(i));

}

printf("\n");

return 0;

}

A screenshot of a computer

Description automatically generated

**MODULE-3**

1. **For loop in C**

#include <stdio.h>  
#include <string.h>  
#include <math.h>  
#include <stdlib.h>  
int main()  
{  
    int a, b,i;  
    scanf("%d\n%d", &a, &b);  
  // Complete the code.  
    for(i=a;i<=b;i++)  
      {  
          if(i<10)  
                        {  
              if(i==1)  
              printf("one\n");  
              else if(i==2)  
              printf("two\n");  
              else if(i==3)  
              printf("three\n");  
              else if(i==4)  
              printf("four\n");  
              else if(i==5)  
              printf("five\n");  
              else if(i==6)  
              printf("six\n");  
              else if(i==7)  
              printf("seven\n");  
              else if(i==8)  
              printf("eight\n");  
              else if(i==9)  
              printf("nine\n");  
          }  
          else {  
              if(i%2==1)  
              printf("odd\n");  
              else {  
              printf("even\n");  
              }  
                    }  
      }  
             return 0;  
}

A screenshot of a computer

Description automatically generated

1. **Calculate Nth term**

#include <stdio.h>

#include <string.h> #include <math.h>

#include <stdlib.h>

//Complete the following function.

int find\_nth\_term(int n, int a, int b, int c) {

//Write your code here.

int term, t1 = a, t2 = b, t3 = c;

if (n == 1)

term = t1;

else if (n == 2)

term = t2;

else if (n == 3)

term = t3;

else {

for (int i = 4; i <= n; i++) {

term = t1 + t2 + t3;

t1 = t2;

t2 = t3;

t3 = term;

}

}

return term;

}

int main() {

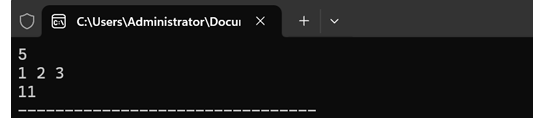
int n, a, b, c;

scanf("%d %d %d %d", Cn, Ca, Cb, Cc); int ans = find\_nth\_term(n, a, b, c);

print("%d", ans);

return 0;

}



1. **Student Marks Sum**

#include<stdio.h>

int main() {

int n, i;

float marks, sum = 0;

// Input the number of students

printf("Enter the number of students: ");

scanf("%d", &n);

// Input the marks for each student and calculate the sum

for (i = 1; i <= n; i++) {

printf("Enter marks for student %d: ", i);

scanf("%f", &marks);

sum += marks;

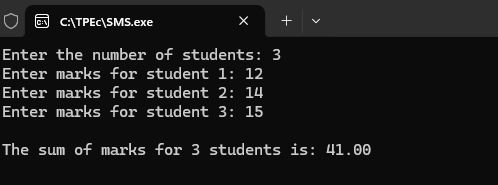
}

// Display the total sum of marks

printf("\nThe sum of marks for %d students is: %.2f\n", n, sum);

return 0;

}

****

1. **Variadic Functions**

#include <stdio.h>

#include <stdarg.h>

// Variadic function to calculate the sum of given integers

int sum(int count, ...) {

va\_list args;

int total = 0;

int i;

// Initialize the argument list

va\_start(args, count);

// Loop through all the arguments

for (i = 0; i < count; i++) {

total += va\_arg(args, int); // Retrieve the next argument

}

// Clean up the argument list

va\_end(args);

return total;

}

int main() {

// Example usage of the sum function

int result1 = sum(3, 10, 20, 30); // Sum of 3 numbers: 10, 20, 30

int result2 = sum(5, 1, 2, 3, 4, 5); // Sum of 5 numbers: 1, 2, 3, 4, 5

// Display the results

printf("The sum of 10, 20, 30 is: %d\n", result1);

printf("The sum of 1, 2, 3, 4, 5 is: %d\n", result2);

return 0;

}

**A black and white screen with white text

Description automatically generated**

1. **Nth Tribonacci Numbers**

#include <stdio.h>

// Function to calculate the Nth Tribonacci number

int tribonacci(int n) {

if (n == 0) return 0;

if (n == 1 || n == 2) return 1;

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

for (int i = 3; i <= n; i++) {

next = a + b + c; // Calculate the next term

a = b; // Update a to the next term

b = c; // Update b to the next term

c = next; // Update c to the next term

}

return c;

}

int main() {

int n;

// Input the value of N

printf("Enter the value of N: ");

scanf("%d", &n);

// Calculate and display the Nth Tribonacci number

printf("The %dth Tribonacci number is: %d\n", n, tribonacci(n));

return 0;

}

**A screenshot of a computer

Description automatically generated**