KPIT Technologies weekly assignment, (NHCE)

Week -2

Name : Kishore Kumar K S ,

USN :1NH20EC065.

Branch : Circuit

Peer Evaluation, Yashavant Kanetkar Book

Book : Let us C , 51- 110 questions

Date : 31/05/2024

51.

Write a program for a matchstick game being played between the computer and a user. Your program should ensure that the computer always wins. Rules for the game are as follows: \_x0010\_ There are 21 matchsticks.

\_x0010\_ The computer asks the player to pick 1, 2, 3, or 4 matchsticks.

\_x0010\_ After the person picks, the computer does its picking.

\_x0010\_ Whoever is forced to pick up the last matchstick loses the game.

#include <stdio.h>

void matchstick\_game() {

int total\_matchsticks = 21;

int player\_pick, computer\_pick;

printf("Welcome to the Matchstick Game!\n");

printf("Rules: There are 21 matchsticks. You can pick 1, 2, 3, or 4 matchsticks.\n");

printf("Whoever is forced to pick the last matchstick loses the game.\n\n");

while (total\_matchsticks > 1) {

// Player's turn

do {

printf("Pick 1, 2, 3, or 4 matchsticks: ");

scanf("%d", &player\_pick);

if (player\_pick < 1 || player\_pick > 4) {

printf("Invalid input. Please pick 1, 2, 3, or 4 matchsticks.\n");

}

} while (player\_pick < 1 || player\_pick > 4);

total\_matchsticks -= player\_pick;

printf("You picked %d matchstick(s). %d matchstick(s) remaining.\n\n", player\_pick, total\_matchsticks);

if (total\_matchsticks == 1) {

printf("You are forced to pick the last matchstick. You lose!\n");

break;

}

// Computer's turn

computer\_pick = 5 - player\_pick;

total\_matchsticks -= computer\_pick;

printf("Computer picks %d matchstick(s). %d matchstick(s) remaining.\n\n", computer\_pick, total\_matchsticks);

if (total\_matchsticks == 1) {

printf("Computer is forced to pick the last matchstick. Computer loses!\n");

break;

}

}

printf("Game over!\n");

}

int main() {

matchstick\_game();

return 0;

}

52

Write a program to enter numbers till the user wants. At the end it should display the count of positive, negative and zeros entered.

#include <stdio.h>

int main() {

int number;

int positive\_count = 0;

int negative\_count = 0;

int zero\_count = 0;

char choice;

do {

printf("Enter a number: ");

scanf("%d", &number);

if (number > 0) {

positive\_count++;

} else if (number < 0) {

negative\_count++;

} else {

zero\_count++;

}

printf("Do you want to enter another number? (y/n): ");

scanf(" %c", &choice);

} while (choice == 'y' || choice == 'Y');

printf("\nCount of positive numbers: %d\n", positive\_count);

printf("Count of negative numbers: %d\n", negative\_count);

printf("Count of zeros: %d\n", zero\_count);

return 0;

}

1. Write a program to receive an integer and find its octal equivalent. (Hint: To obtain octal equivalent of an integer, divide it continuously by 8 till dividend doesn’t become zero, then write the remainders obtained in reverse direction.)

#include <stdio.h>

void find\_octal(int number) {

int octal[32]; // Array to store octal digits

int index = 0; // Index for the array

// Handle the special case when the number is 0

if (number == 0) {

printf("Octal equivalent: 0\n");

return;

}

// Process the number and store remainders in reverse order

while (number > 0) {

octal[index] = number % 8; // Store remainder

number = number / 8; // Update number

index++;

}

// Print the octal digits in reverse order

printf("Octal equivalent: ");

for (int i = index - 1; i >= 0; i--) {

printf("%d", octal[i]);

}

printf("\n");

}

int main() {

int number;

// Get user input

printf("Enter an integer: ");

scanf("%d", &number);

// Find and display the octal equivalent

find\_octal(number);

return 0;

}

1. Write a program to find the range of a set of numbers entered through the keyboard. Range is the difference between the smallest and biggest number in the list.

#include <stdio.h>

int main() {

int n, i;

float num, smallest, largest;

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

scanf("%d", &n);

printf("Enter number 1: ");

scanf("%f", &num);

smallest = num;

largest = num;

for (i = 2; i <= n; i++) {

printf("Enter number %d: ", i);

scanf("%f", &num);

if (num < smallest) {

smallest = num;

}

if (num > largest) {

largest = num;

}

}

printf("Range of the entered numbers: %.2f\n", largest - smallest);

return 0;

}

1. Calculation of simple interest for 3 sets of p, n and r

# include <stdio.h>

int main( )

{ int p, n, count ;

float r, si ;

for ( count = 1 ; count <= 3 ; count = count + 1 )

{

printf ( "Enter values of p, n, and r " ) ;

scanf ( "%d %d %f", &p, &n, &r ) ;

si = p \* n \* r / 100 ;

printf ( "Simple Interest = Rs.%f\n", si ) ;

}

Return 0;

}

1. Nested loops

/\* Demonstration of nested loops \*/

# include<stdio.h>

int main( )

{ int r, c, sum ;

for ( r = 1 ; r <= 3 ; r++ ) /\* outer loop \*/

{ for ( c = 1 ; c <= 2 ; c++ ) /\* inner loop \*/

{ sum = r + c ;

printf ( "r = %d c = %d sum = %d\n", r, c, sum ) ;

} } return 0 ; }

1. Write a program to determine whether a number is prime or not. A prime number is said to be prime if it is divisible only by 1 or itself.

# include<stdio.h>

int main( )

{ int num, i ;

printf ( "Enter a number " ) ;

scanf ( "%d", &num ) ; i = 2 ;

while ( i <= num - 1 )

{

if ( num % i == 0 ) {

printf ( "Not a prime number\n" ) ;

break ;

}

i++ ;

}

if ( i == num )

printf ( "Prime number\n" ) ; }

1. A continue statement

# include <stdio.h>

int main( )

{ int i, j ;

for ( i = 1 ; i <= 2 ; i++ )

{

for ( j = 1 ; j <= 2 ; j++ )

{

if ( i == j )

continue ;

printf ( "%d %d\n", i, j ) ;

} }

return 0 ;

}

/\* Execution of a loop an unknown number of times \*/

# include <stdio.h>

int main( )

{

char another ;

int num ;

Do

{

printf ( "Enter a number " ) ;

scanf ( "%d", &num ) ;

printf ( "square of %d is %d\n", num, num \* num ) ;

printf ( "Want to enter another number y/n " ) ;

fflush ( stdin ) ;

scanf ( "%c", &another ) ;

}

while ( another == 'y' ) ;

return 0 ;

}

Enter a number 5

square of 5 is 25

Want to enter another number y/n y

Enter a number 7 square of 7 is 49

Want to enter another number y/n n

1. Write a program to print all prime numbers from 1 to 300.

Program

/\* Generate all prime numbers from 1 to 300 \*/

# include <stdio.h>

int main( )

{ int i, n = 1 ;

printf ( "\nPrime numbers between 1 and 300 are :\n1\t" ) ;

for ( n = 1 ; n <= 300 ; n++ )

{ i = 2 ;

for ( i = 2 ; i < n ; i++ )

{

if ( n % i == 0 )

break ;

}

if ( i == n )

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

}

return 0 ;

}

1. Write a program to add first seven terms of the following series using a for loop. 1/ 1!+ 2/ 2!+ 3/ 3! ……

Program /\* Sum of first seven terms of a series \*/

# include<stdio.h>

int main( )

{ int i = 1, j ;

float fact, sum = 0.0 ;

for ( i = 1 ; i <= 7 ; i++ )

{ fact = 1.0 ;

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

fact = fact \* j ;

sum = sum + i / fact ;

} printf ( "Sum of series = %f\n", sum ) ;

return 0 ; }

1. Write a program to generate all combinations of 1, 2 and 3 using for loop. Program

/\* Generate all possible combinations of 1 2 3 \*/

# include <stdio.h>

int main( )

{ int i = 1, j = 1, k = 1 ;

for ( i = 1 ; i <= 3 ; i++ )

{ for ( j = 1 ; j <= 3 ; j++ )

{ for ( k = 1 ; k <= 3 ; k++ )

printf ( "%d %d %d\n", i , j , k ) ;

} }

return 0 ;

}

1. (a) The break statement is used to exit from:
2. An if statement

2. A for loop

3. A program

4. The main( ) function

Ans:A for loop

1. b) A do-while loop is useful when we want that the statements within the loop must be executed:

1. Only once

2. At least once

3. More than once

4. None of the above

Ans: Atleast once

1. In what sequence the initialization, testing and execution of body is done in a do-while loop?
2. Initialization, execution of body, testing
3. . Execution of body, initialization, testing
4. . Initialization, testing, execution of body
5. . None of the above

Ans: 3. Initialization, testing, execution of body

65 .Which of the following is not an infinite loop?

1. int i = 1 ; while ( 1 ) { i++ ; }
2. . for ( ; ; ) ;
3. int t = 0, f ; while ( t ) { f = 1 ;
4. int y, x = 0 ; do { y = x ;

Ans; 1.This loop will continue indefinitely because the condition while (1) is always true.

66.Which of the following statements is true for the following program?

# include<stdio.h>

int main( )

{ int x = 10, y = 100 % 90 ;

for ( i = 1 ; i <= 10 ; i++ ) ;

if ( x != y ) ;

printf ( "x = %d y = %d\n", x, y ) ;

return 0 ; }

1. The printf( ) function is called 10 times.
2. The program will produce the output x = 10 y = 10.
3. The ; after the if ( x != y ) will not produce an error.
4. . The program will not produce any output. 5. The printf( ) function is called infinite times.

Ans: option 3 , The semicolon after the if (x != y) statement creates an empty statement that does nothing. It will not produce any error, but it's also irrelevant to the program's logic. So, option 3 is correct.

1. Which of the following statement is true about a for loop used in a C program?

1. for loop works faster than a while loop.

1. All things that can be done using a for loop can also be done using a while loop.
2. for ( ; ; ) implements an infinite loop.
3. for loop can be used if we want statements in a loop to get executed at least once.
4. for loop works faster than a do-while loop.

Ans: option 2,3,4 is correct

1. Write a program to print the multiplication table of the number entered by the user. The table should get displayed in the following form:

29 \* 1 = 29

29 \* 2 = 58

#include <stdio.h>

int main() { int number, limit; // Input the number from the user

printf("Enter the number: ");

scanf("%d", &number); // Input the limit for the multiplication table printf("Enter the limit for the multiplication table: ");

scanf("%d", &limit); // Print the multiplication table

for (int i = 1; i <= limit; i++)

{

printf("%d \* %d = %d\n", number, i, number \* i);

}

return 0; }

1. switch cases :

# include<stdio.h>

int main( )

{ int i = 2 ;

switch ( i ) {

case 1 :

printf ( "I am in case 1 \n" ) ;

case 2 :

printf ( "I am in case 2 \n" ) ;

case 3 :

printf ( "I am in case 3 \n" ) ;

default :

printf ( "I am in default \n" ) ;

} return 0 ; }

70 .Switch and break

# include<stdio.h>

int main( )

{ int i = 2 ; switch ( i )

{ case 1 :

printf ( "I am in case 1 \n" ) ;

break ;

case 2 :

printf ( "I am in case 2 \n" ) ;

break ;

case 3 :

printf ( "I am in case 3 \n" ) ;

break ;

default :

printf ( "I am in default \n" ) ; }

return 0 ; }

71 .goto statements

# include<stdio.h>

# include<stdlib.h>

int main( )

{ int goals ;

printf ( "Enter the number of goals scored against India" ) ;

scanf ( "%d", &goals ) ;

if ( goals <= 5 )

goto sos ;

else {

printf ( "About time soccer players learnt C\n" ) ;

printf ( "and said goodbye to soccer\n" ) ;

exit ( 1 ) ; /\* terminates program execution \*/ }

sos :

printf ( "To err is human!\n" ) ;

return 0 ;

}

72.what will be output

# include<stdio.h>

int main( )

{

char suite = 3 ;

switch ( suite )

{

case 1 :

printf ( "Diamond\n" ) ;

case 2 :

printf ( "Spade\n" ) ;

default :

printf ( "Heart\n" ) ; }

printf ( "I thought one wears a suite\n" ) ;

return 0 ; }

Output: Heart

I thought one wears a suite

1. ) # include

int main( ) {

int c = 3 ; switch ( c )

{ case '3' :

printf ( "You never win the silver prize\n" ) ;

break ;

case 3 :

printf ( "You always lose the gold prize\n" ) ;

break ;

default :

printf ( "Of course provided you win a prize\n" ) ;

}

return 0 ; }

Output:You always lose the gold prize

74 .# include<stdio.h>

int main( )

{ int i = 3 ;

switch ( i )

{ case 0 :

printf ( "Customers are dicey\n" ) ;

case 1 + 2 :

printf ( "Markets are pricey\n" ) ;

case 4 / 2 :

printf ( "Investors are moody\n" ) ;

} return 0 ; }

output:Markets are pricey

Investors are moody

75 .# include<stdio.h>

int main( )

{ int k ;

float j = 2.0 ;

switch ( k = j + 1 ) {

case 3 :

printf ( "Trapped\n" ) ;

break ;

default :

printf ( "Caught!\n" ) ;

return 0 ; }

Output:Trapped

1. # include <stdio.h>

int main( )

{ int ch = 'a' + 'b' ;

switch ( ch )

{ case 'a' : case 'b' :

printf ( "Look at 10 ideas, 11th will occur to you\n" ) ;

case 'A' :

printf ( "If you have a good idea, project it\n" ) ;

case 'b' + 'a' :

printf ( "Have ideas, will fly\n" ) ;

} return 0 ; }

Output:Have ideas, will fly

1. Point out the errors, if any, in the following programs:

(a) # include<stdio.h>

int main( )

{ int suite = 1 ;

switch ( suite ) ;

{

case 0 ;

printf ( "Club\n" ) ;

case 1 ;

printf ( "Diamond\n" ) ;

} return 0 ; }

Ans: **Semicolon after the switch statement**

**Incorrect syntax for case labels**:

1. # include<stdio.h>

int main( ) {

int temp ;

scanf ( "%d", &temp ) ;

switch ( temp )

{ case ( temp <= 20 ) :

printf ( "Ooooooohhhh! Damn cool!\n" ) ;

case ( temp > 20 && temp <= 30 ) :

printf ( "Rain rain here again!\n" ) ;

case ( temp > 30 && temp <= 40 ) :

printf ( "Wish I am on Everest\n" ) ;

default :

printf ( "Good old Nagpur weather\n" ) ;

} return 0 ; }

Output: **Incorrect usage of** switch **with expressions**

**Missing** break **statements**

**Default case placement**

1. # include <stdio.h>

int main( )

{ float a = 3.5 ;

switch ( a )

{ case 0.5 :

printf ( "The art of C\n" ) ;

break ;

case 1.5 :

printf ( "The spirit of C\n" ) ;

break ; case 2.5 :

printf ( "See through C\n" ) ;

break ; } return 0 ; }

Ans: Using switch with a float variable

Case labels with floating-point value

1. ) # include<stdio.h>

int main( )

{ int a = 3, b = 4, c ; c = b – a ;

switch ( c ) {

case 1 || 2 :

printf ( "God give me a chance to change things\n" ) ;

break ;

case a || b :

printf ( "God give me a chance to run my show\n" ) ;

break ; } return 0 ;

Ans: **Incorrect syntax for case labels**: In a switch statement, case labels must be constant expressions, not logical expressions like 1 || 2 or variables like a || b.

· **Confusion between** switch **statement and** if **statement logic**: the syntax case 1 || 2 is not valid in C. Similarly, case a || b is also not valid. Case labels should be constants or compile-time expressions.

1. Usage of functions

# include<stdio.h>

void message( ) ;

/\* function prototype declaration \*/

int main( )

{ message( ) ;

/\* function call \*/

printf ( "Cry, and you stop the monotony!\n" ) ;

return 0 ;

} void message( ) /\* function definition \*/

{ printf ( "Smile, and the world smiles with you...\n" ) ; }

Output : Smile, and the world smiles with you... Cry, and you stop the monotony!

1. calling functions

# include<stdio.h>

void italy( ) ; void brazil( ) ; void argentina( ) ;

int main( )

{ printf ( "I am in main\n" ) ;

italy( ) ;

brazil( ) ;

argentina( ) ;

return 0 ; }

void italy( ) {

printf ( "I am in italy\n" ) ; }

void brazil( )

{ printf ( "I am in brazil\n" ) ;

} void argentina( ) {

printf ( "I am in argentina\n" ) ; }

The output : I am in main

I am in italy

I am in brazil

I am in argentina

1. Sending and receiving values between functions

# include<stdio.h>

int calsum ( int x, int y, int z ) ;

int main( ) {

int a, b, c, sum ;

printf ( "Enter any three numbers " ) ;

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

sum = calsum ( a, b, c ) ;

printf ( "Sum = %d\n", sum ) ;

return 0 ; }

int calsum ( int x, int y, int z ) {

int d ; d = x + y + z ;

return ( d ) ; }

The output of the program..

Enter any three numbers 10 20 30

Sum = 60

1. When value of formal argument is changed

# include<stdio.h>

void fun ( int ) ;

int main( ) {

int a = 30 ; fun ( a ) ;

printf ( "%d\n", a ) ;

return 0 ; }

void fun ( int b ) {

b = 60 ;

printf ( "%d\n", b ) ; }

The output : 60 30

1. Usage of library functions

# include<stdio.h>

# include<stdlib.h>

int main( )

{ float a = 0.5 ;

float w, x, y, z ;

w = sin ( a ) ;

x = cos ( a ) ;

y = tan ( a ) ;

z = pow ( a, 2 ) ;

printf ( "%f %f %f %f\n", w, x, y, z ) ;

return 0 ; }

86 . Return type of functions

# include<stdio.h>

float square ( float ) ;

int main( ) {

float a, b ;

printf ( "Enter any number " ) ;

scanf ( "%f", &a ) ; b = square ( a ) ;

printf ( "Square of %f is %f\n", a, b ) ;

return 0 ; }

float square ( float x ) {

float y ; y = x \* x ;

return ( y ) ; }

sample runs of this program…

Enter any number 3

Square of 3 is 9.000000

Enter any number 1.5

Square of 1.5 is 2.250000

Enter any number 2.5

Square of 2.5 is 6.250000

87.Write a function to calculate the factorial value of any integer entered through the keyboard.

/\* Calculate factorial value of an integer using a function \*/

# include<stdio.h>

int fact ( int ) ;

int main( ) {

int num ; int factorial ;

printf ( "\nEnter a number: " ) ;

scanf ( "%d", &num ) ;

factorial = fact ( num ) ;

printf ( "Factorial of %d = %ld\n", num, factorial ) ;

return 0 ; }

int fact ( int num ) {

int i ; int factorial = 1 ;

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

factorial = factorial \* i ;

return ( factorial ) ; }

Output : Enter a number: 6

Factorial of 6 = 720

1. Write a function power ( a, b ), to calculate the value of a raised to b. /\* Program to calculate power of a value \*/

# include<stdio.h>

float power ( float, int ) ;

int main( ) { float x, pow ; int y ;

printf ( "\nEnter two numbers: " ) ;

scanf ( "%f %d", &x, &y ) ;

pow = power ( x , y ) ;

printf ( "%f to the power %d = %f\n", x, y, pow ) ;

return 0 ;

} float power ( float x, int y ) {

int i ; float p = 1 ;

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

p = p \* x ; return ( p ) ; }

Output :Enter two numbers: 1.5 3

1.500000 to the power 3 = 3.375000

1. Define a function to convert any given year into its Roman equivalent. Use these Roman equivalents for decimal numbers: 1 – I, 5 – V, 10 – X, 50 – L, 100 – C, 500 – D, 1000 – M. Example: Roman equivalent of 1988 is mdcccclxxxviii. Roman equivalent of 1525 is mdxxv.

# include<stdio.h>

int romanise ( int, int, char ) ;

int main( )

{ int yr ;

printf ( "\nEnter year: " ) ;

scanf ( "%d", &yr ) ;

yr = romanise ( yr, 1000, 'm' ) ;

yr = romanise ( yr, 500, 'd' ) ;

yr = romanise ( yr, 100, 'c' ) ;

yr = romanise ( yr, 50, 'l' ) ;

yr = romanise ( yr, 10, 'x' ) ;

yr = romanise ( yr, 5, 'v' ) ;

romanise ( yr, 1, 'i' ) ; return 0 ; }

int romanise ( int y, int k, char ch )

{ int i, j ; j = y / k ;

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

printf ( "%c", ch ) ; return ( y % k ) ; }

Output :Enter year:1988

Mdcccclxxxviii

1. point the errors
2. # include<stdio.h>

int addmult ( int, int )

int main( ) {

int i = 3, j = 4, k, l ;

k = addmult ( i, j ) ;

l = addmult ( i, j ) ;

printf ( "%d %d\n", k, l ) ;

return 0 ; }

int addmult ( int x, int y )

{ int u, v ; u = x + y ; v = x \* y ; return ( u, v ) ; }

Ans:**Function Declaration**: The function addmult is declared before it is defined.

**Syntax in Function Declaration**: The function addmult lacks parameter names in its declaration.

**Return Statement in** addmult **Function**: The addmult function is returning (u, v), which actually evaluates u, discards it, and then returns v.

1. # include<stdio.h>

int main( ) {

int a ; a = message( ) ;

return 0 ;

}

void message( ) {

printf ( "Learn from him online at ykanetkar.com\n" ) ;

return ; }

Ans:**Function Declaration**: The function message is called before it is declared

**Function Return Type Mismatch**: The message function is declared as void

1. # include<stdio.h>

int main( ) {

float a = 15.5 ;

char ch = 'C' ;

printit ( a, ch ) ;

return 0 ; }

printit ( a, ch ) {

printf ( "%f %c\n", a, ch ) ; }

Ans:**Function Declaration**: The function printit is called before it is declared.

**Function Declaration Syntax**: The function printit lacks a return type in its declaration

1. # include<stdio.h>

int main( )

{ let\_us\_c( ) {

printf ( "Learn C online…\n" ) ;

printf ( "At ykanetkar.com\n" ) ;

} return 0 ; }

Ans:**Function Declaration**: The function printit is called before it is declared.

**Function Declaration Syntax**: The function printit lacks a return type in its declaration

1. Function call by value

# include<stdio.h>

void swapv ( int x, int y ) ;

int main( ) { int a = 10, b = 20 ;

swapv ( a, b ) ;

printf ( "a = %d b = %d\n", a, b ) ;

return 0 ; }

void swapv ( int x, int y )

{ int t ; t = x ; x = y ; y = t ;

printf ( "x = %d y = %d\n", x, y ) ; }

The output : x = 20 y = 10

a = 10 b = 20

1. Call by reference

# include<stdio.h>

void swapr ( int \*, int \* ) ;

int main( ) {

int a = 10, b = 20 ;

swapr ( &a, &b ) ;

printf ( "a = %d b = %d\n", a, b ) ;

return 0 ; }

void swapr ( int \*x, int \*y ) {

int t ; t = \*x ; \*x = \*y ; \*y = t ; }

The output : a = 20 b = 10

1. Write a function that receives 5 integers and returns the sum, average and standard deviation of these numbers. Call this function from main( ) and print the results in main( ).

Program /\* Function which returns sum, average and standard deviation \*/

# include<stdio.h>

# include<math.h>

void stats ( int \*, int \*, double \* ) ;

int main( ) { int sum, avg ; double stdev ;

stats ( ∑, &avg, &stdev ) ;

printf ( "Sum = %d \nAverage = %d \nStandard deviation = %lf\n", sum, avg, stdev ) ;

return 0 ; }

void stats ( int \*sum, int \*avg, double \*stdev ) { int n1, n2, n3, n4, n5 ;

printf ( "\nEnter 5 numbers: " ) ; scanf ( "%d%d%d%d%d", &n1, &n2, &n3, &n4, &n5 ) ;

\*sum = n1 + n2 + n3 + n4 + n5 ;

\*avg = \*sum / 5 ;

\*stdev = sqrt ( ( pow ( ( n1 - \*avg ), 2.0 ) + pow ( ( n2 - \*avg ), 2.0 ) + \ pow ( ( n3 - \*avg ), 2.0 ) + pow ( ( n4 - \*avg ), 2.0 ) + \ pow ( ( n5 - \*avg ), 2.0 ) ) / 4 ) ; }

Output: Enter 5 numbers: 10 20 30 40 50

Sum = 150

Average = 30

Standard deviation = 15.811388

1. Write a program that defines a function that calculates power of one number raised to another and factorial value of a number in one call.

# include<stdio.>

void power\_fact ( float, int, int, float \*, int \* ) ;

int main( ) {

float a ; int b, number, factorial ;

float pow ;

printf ( "Enter a and b for calculating a raised to b: " ) ;

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

printf ( "Enter number whose factorial is to be calculated: " ) ;

scanf ( "%d", &number ) ;

power\_fact ( a, b, number, &pow, &factorial ) ;

printf ( "Power = %f Factorial = %d", pow, factorial ) ;

return 0 ; } void power\_fact ( float x, int y, int num, float \*power, int \*fact ) { float res = 1 ; int i ;

for ( i = 1 ; i <= y ; i++ ) res = res \* x ;

\*power = res ; res = 1 ;

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

res = res \* i ;

\*fact = res ; }

Output: Enter a and b for calculating a raised to b:2 5

Enter number whose factorial is to be calculated: 6

Power = 32.000000 Factorial = 720

1. What is output :

# include<stdio.h

void fun ( int, int ) ;

int main( ) {

int i = 5, j = 2 ;

fun ( i, j ) ;

printf ( "%d %d\n", i, j ) ;

return 0 ; }

void fun ( int i, int j ) {

i = i \* i ; j = j \* j ; }

Output :5 2

1. # include<stdio.h>

void fun ( int \*, int \* ) ;

int main( ) {

int i = 5, j = 2 ;

fun ( &i, &j ) ;

printf ( "%d %d\n", i, j ) ;

return 0 ; }

void fun ( int \*i, int \*j ) {

\*i = \*i \* \*i ; \*j = \*j \* \*j ; }

Output : 25 4

1. # include<stdio.h>

int main( ) {

float a = 13.5 ;

float \*b, \*c ; b = &a ;

/\* suppose address of a is 1006 \*/

c = b ;

printf ( "%u %u %u\n", &a, b, c ) ;

printf ( "%f %f %f %f %f\n", a, \*(&a), \*&a, \*b, \*c ) ;

return 0 ; }

Output:

13.500000 13.500000 13.500000 13.500000 13.500000