1. Write a program to calculate x^n without using library function pow( ) but using user defined function.

#include<stdio.h>

int power(int , int );

int main()

{

int x,y;

printf("enter the value of x\n");

scanf("%d",&x);

printf("enter the value of n\n");

scanf("%d",&y);

power(x,y);

return 0;

}

int power(int x,int y)

{

int power=1;

while(y>0)

{

power=power\*x;

y--;

;

}

printf("the result of the number=%d",power);

return power;

}

1. Write a program that input the meal charge of a customer. The tax should be 20% of the meal cost. The tip should be 15% of the total after adding the tax. Display the total bill on the screen using function.

#include<stdio.h>

int bill(int);

int main()

{

int m;

printf("enter the meal charge of customer\n");

scanf("%d",&m);

bill(m);

return 0;

}

bill(int m)

{

int t,r;

t=m+((m\*20)/100);

r=t+((15\*t)/100);

printf("total bill to be paid by customer=%d",r);

return r;

}

3.Write a program to input coefficients of quadratic equation and pass them to function( ) QUAD. This returnable function computes whether roots of a quadratic equation are real or imaginary.

#include<stdio.h>

int quad(int,int,int);

int main()

{

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

printf("enter the coefficients of quadratic equation\n");

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

quad(a,b,c);

return 0;

}

quad(int a,int b,int c)

{

int d;

d=(b\*b)-4\*a\*c;

if(d>=0)

printf("the roots are real");

else

printf("the roots are imaginary");

return d;

}

1. Electricity Bill Statement (EBS) takes units consumed from consumer and calculates electricity charges (EC) using provided criteria:

1 – 100 units @ Rs. 2.00/- (per unit)

101 – 200 units @ Rs. 3.50/- (per unit)

201 and more units @ Rs. 4.50/- (per unit)

General sale tax which is the 10% of the EC. Amount due (EC + Gen. Sale tax) .

#include<stdio.h>

int ebs(int);

int main()

{

int a;

printf("enter the unit consumed\n");

scanf("%d",&a);

ebs(a);

return 0;

}

ebs(int a)

{

float r,t;

if(a<100)

{

r=a\*2;

}

else if(a<=200)

{

r=200+((a-100)\*3.50);

}

else

{

r=550+((a-200)\*4.50);

}

t=r+((r\*10)/100);

printf("total due amount is %f",t);

return t;

}

1. Write a program to swap two numbers using call by value.

#include <stdio.h>

void swap(int , int); //prototype of the function

int main()

{

int a = 10;

int b = 20;

printf("Before swapping the values in main a = %d, b = %d\n",a,b);

swap(a,b);

printf("After swapping values in main a = %d, b = %d\n",a,b);

}

void swap (int a, int b)

{

int temp;

temp = a;

a=b;

b=temp;

printf("After swapping values in function a = %d, b = %d\n",a,b);

}

1. Define function. What are the types of function in c? Categorize user defined functions.

A function is a **group of statements that together perform a task**. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions. ... A function declaration tells the compiler about a function's name, return type, and parameters.

category I: **Functions with no arguments** and no return values. Category 2: Functions with no arguments and with return values. Category 3: Functions with arguments and no return values. Category 4: Functions with arguments and with return values.

12. Discuss the following terms –function declaration, function definition, actual and formal arguments, calling function and called function with suitable example.

a function declaration **tells the compiler about a function's name, return type, and parameters**. A function definition provides the actual body of the function. The C standard library provides numerous built-in functions that your program can call.

The variables declared in the function prototype or definition are known as Formal arguments and the **values that are passed to** the called function from the main function are known as Actual arguments. The actual arguments and formal arguments must match in number, type, and order.

the Function which calls another Function is called **Calling Function** and function which is called by another Function is call **Called Function.**

**int** min (**int** n1, **int** n2); // min() is the name of a function that contains n1 and n2 parameters

1. {
2. // declaration of the local variable.
3. **int** out;
4. **if** ( n1 > n2)
5. out = n1; // return n1 when n1 is greater than n2.
6. **else**
7. out = n2; // return n2 when n2 is greater than n1.
8. **return** out;
9. }
   1. What do you mean by call by value? Give one example.

The **call by value** method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

By default, C programming uses *call by value* to pass arguments. In general, it means the code within a function cannot alter the arguments used to call the function. Consider the function **swap()** definition as follows.

#include <stdio.h>

/\* function declaration \*/

void swap(int x, int y);

int main () {

/\* local variable definition \*/

int a = 100;

int b = 200;

printf("Before swap, value of a : %d\n", a );

printf("Before swap, value of b : %d\n", b );

/\* calling a function to swap the values \*/

swap(a, b);

printf("After swap, value of a : %d\n", a );

printf("After swap, value of b : %d\n", b );

return 0;

}

void swap(int x, int y) {

int temp;

temp = x; /\* save the value of x \*/

x = y; /\* put y into x \*/

y = temp; /\* put temp into y \*/

return;

}