```
/*
1.Function are used to convert a large C program into smaller pieces.
2.A function can be called multiple times to provide reusability and
modularity to the
C program.
3.Also called procedure or subroutine.
Syntax:
return_type function_name(data_type parameter 1){
//Code to be executed
}
Advantages of a function:
1.We can avoid rewriting same logic through functions.
2.We can divide work among programmers through functions.
3.We can easily debug a program using functions.
Declaration, Definition and Call:
1.A function is declared to tell a compiler about its existence.
2.A function is defined to get some task done.
3.A function is called in order to be used.
Types of functions:
1.Library functions:
```

Functions included in C header files.

2.User defined functions:

Functions created by C programmer to reduce complexity of a program.

Function code examples:

1.Without argument and with return type:

```
#include <stdio.h>

void myname(){

printf("Aditya Yadav");

}

int main()

{

printf("My name is: ");

myname();

return 0;

}
```

2.With argument and with return value:

```
#include <stdio.h>

int sum(int a,int b){

return a+b;

}
```

```
int main()

{

int a,b,c;

a=9;

b=87;

c=sum(a,b);

printf("The sum is %d",c);

return 0;

}
```

3.With argument and with return value:

```
#include <stdio.h>

int takenumber(){

int i;

printf("Enter a number:");

scanf("%d",&i);

printf("The number is %d\n",i);

return i;

}

int main()

{
```

```
int c;

c=takenumber();

return 0;

}

4.With argument and without return value:

#include <stdio.h>

void printstar(int n){

for(int i=0;i<n;i++){

printf("%c",'*');

}

}

int main()

{

printstar(7);

return 0;

}

*/

#include <stdio.h>

int sum(int a, int b)

{
```

```
return a + b;

}

int main()

{

int a, b, c;

a = 9;

b = 87;

c = sum(a, b);

printf("The sum is %d", c);

return 0;

}
```