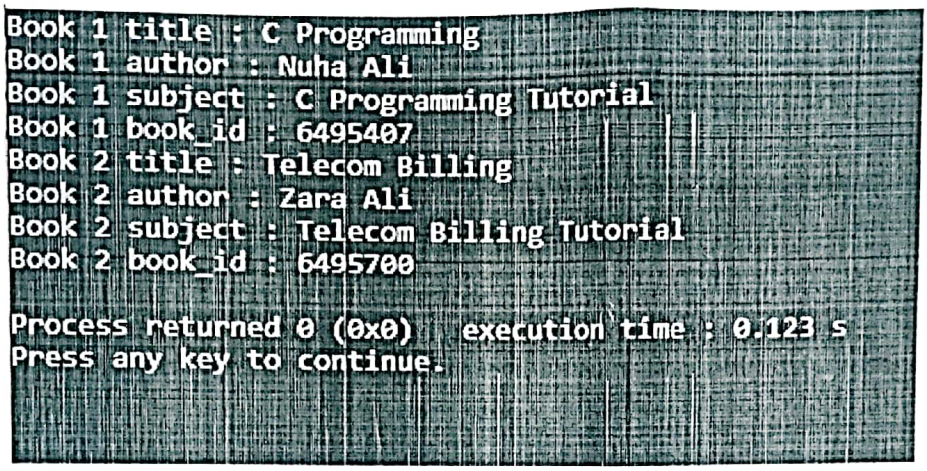
STRUCTURES IN C:

```
#include <stdio.h>
#include <string.h>
struct Books
{
  char  title[50];
  char  author[50];
  char  subject[100];
  int   book_id;
};

int main( )
{
  struct Books Book1;
  struct Books Book2;
  strcpy( Book1.title, "C Programming");
  strcpy( Book1.author, "Nuha Ali");
  strcpy( Book1.subject, "C Programming Tutorial");
  Book1.book_id = 6495407;
  strcpy( Book2.title, "Telecom Billing");
  strcpy( Book2.author, "Zara Ali");
  strcpy( Book2.subject, "Telecom Billing Tutorial");
  Book2.book_id = 6495700;
  printf( "Book 1 title : %s\n", Book1.title);
  printf( "Book 1 author : %s\n", Book1.author);
  printf( "Book 1 subject : %s\n", Book1.subject);
  printf( "Book 1 book_id : %d\n", Book1.book_id);
  printf( "Book 2 title : %s\n", Book2.title);
  printf( "Book 2 author : %s\n", Book2.author);
  printf( "Book 2 subject : %s\n", Book2.subject);
```

```
   printf( "Book 2 book_id : %d\n", Book2.book_id);
   return 0;
}
```

OUTPUT:

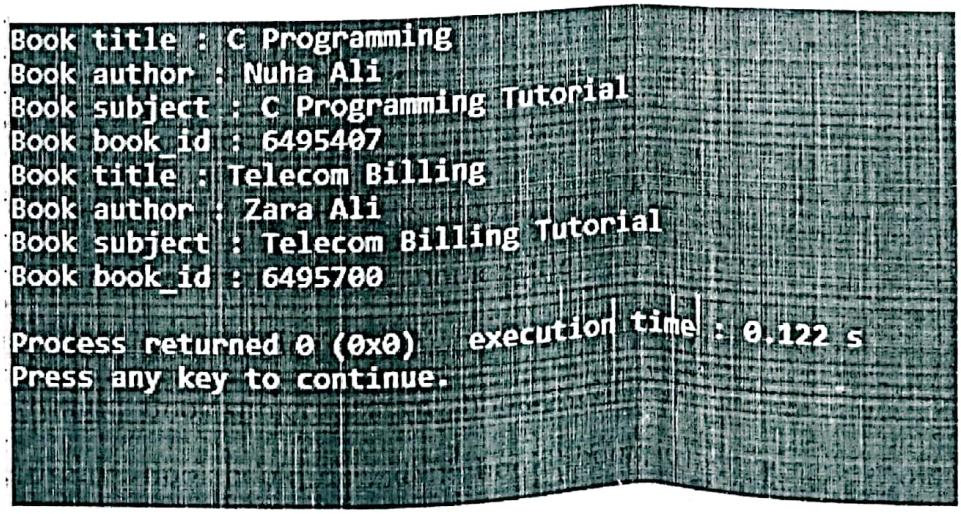


POINTERS TO STRUCTURES:

```
#include <stdio.h>
#include <string.h>
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};
/* function declaration */
void printBook( struct Books *book );
int main( ) {
   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */
   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
```

```
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;
   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;
   /* print Book1 info by passing address of Book1 */
   printBook( &Book1 );
   /* print Book2 info by passing address of Book2 */
   printBook( &Book2 );
   return 0;
}
void printBook( struct Books *book ) {
   printf( "Book title : %s\n", book->title);
   printf( "Book author : %s\n", book->author);
   printf( "Book subject : %s\n", book->subject);
   printf( "Book book_id : %d\n", book->book_id);
}
```

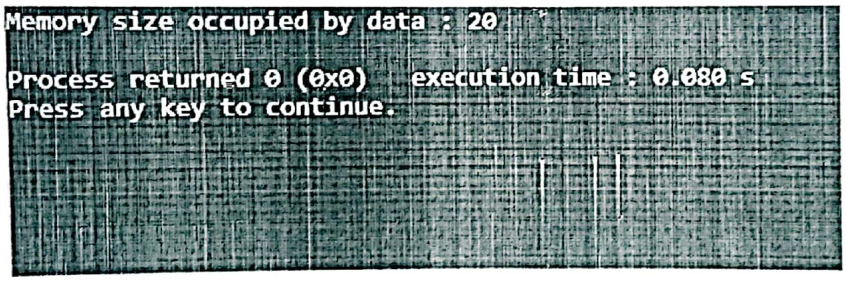
OUTPUT:



C-UNIONS:

```
#include <stdio.h>
#include <string.h>
union Data {
  int i;
  float f;
  char str[20];
};
int main( ) {
   union Data data;
printf( "Memory size occupied by data : %d\n", sizeof(data));
return 0;
}
```

OUTPUT:

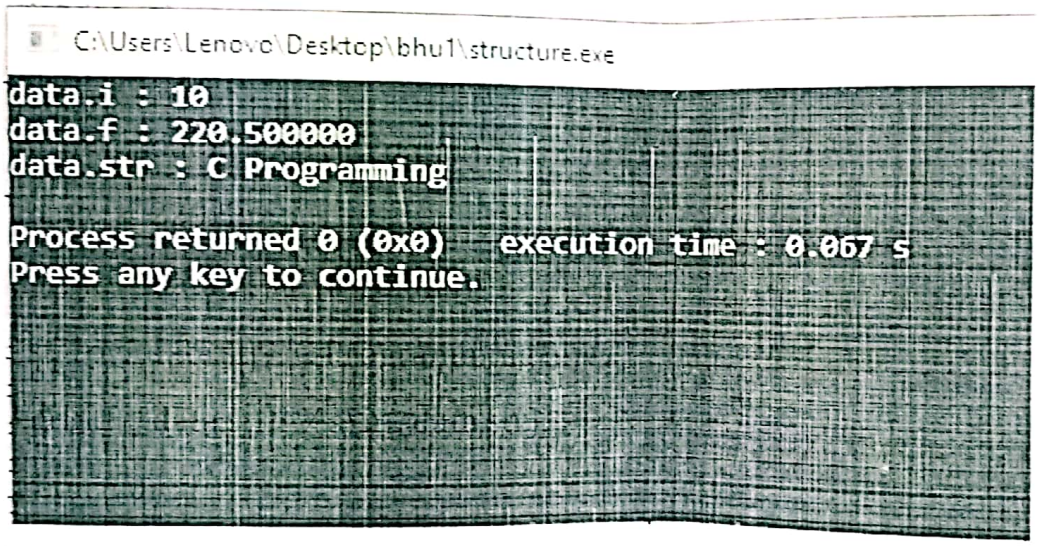


ACCESSING UNION MEMBERS:

```
#include <stdio.h>
#include <string.h>
union Data {
   int i;
   float f;
  char str[20];
};
int main( ) {
    union Data data;
```

```
    data.i = 10;
    printf( "data.i : %d\n", data.i);
    data.f = 220.5;
    printf( "data.f : %f\n", data.f);
    strcpy( data.str, "C Programming");
    printf( "data.str : %s\n", data.str);
   return 0;
}
```

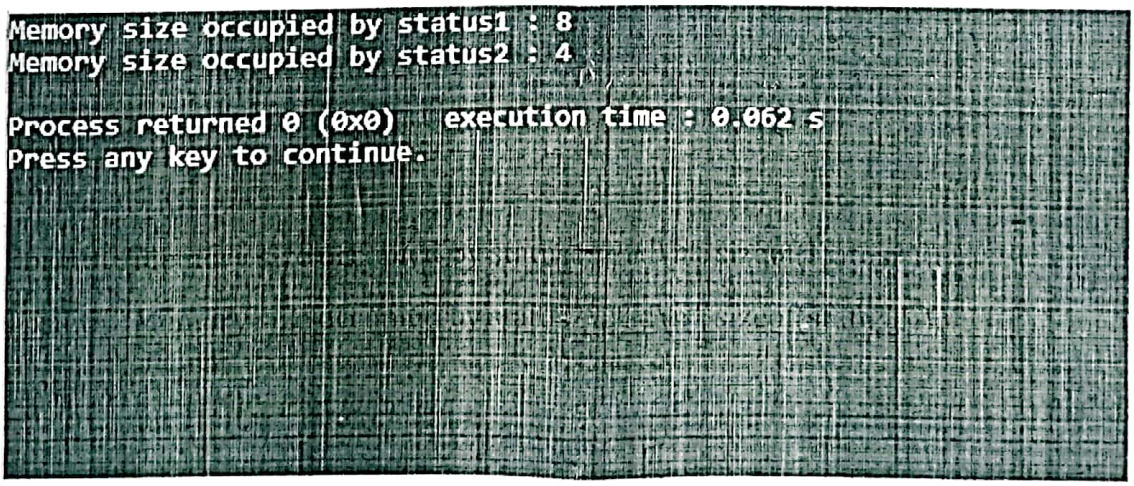
OUTPUT:



C-BITFIELDS:

```
#include <stdio.h>
#include <string.h>
/* define simple structure */
struct {
  unsigned int widthValidated;
  unsigned int heightValidated;
} status1;
/* define a structure with bit fields */
struct {
  unsigned int widthValidated : 1;
  unsigned int heightValidated : 1;
} status2;
 int main( ) {
  printf( "Memory size occupied by status1 : %d\n", sizeof(status1));
  printf( "Memory size occupied by status2 : %d\n", sizeof(status2));
  return 0;
}
```

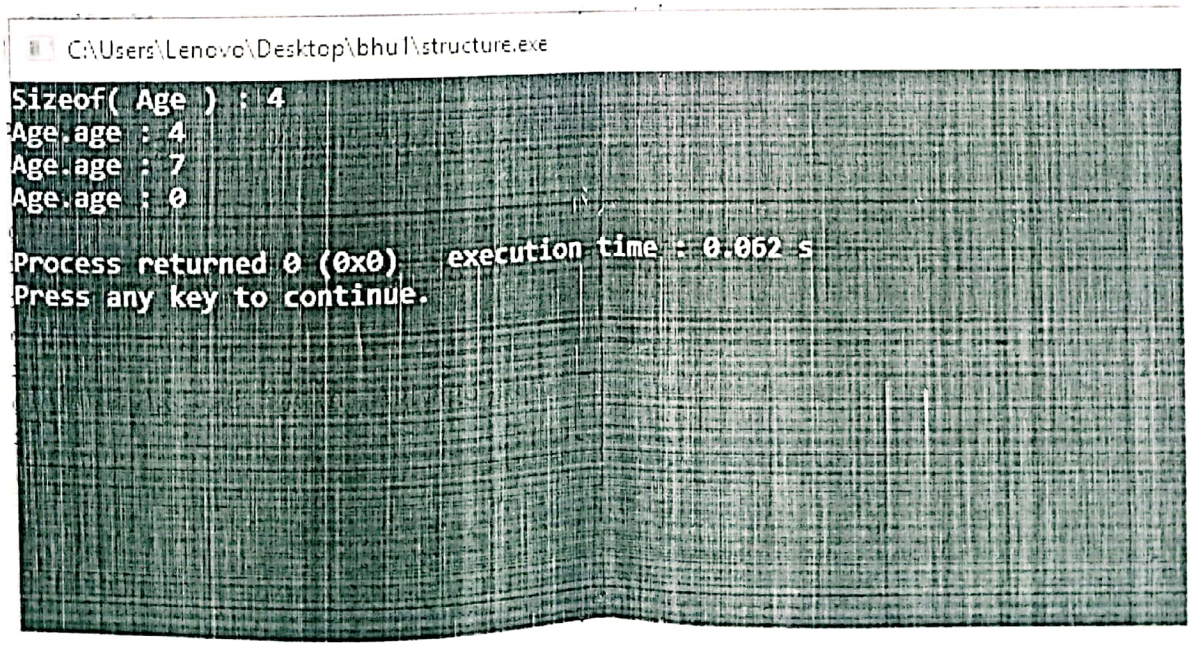
OUTPUT:



BIT FIELD DECLARATION:

```
#include <stdio.h>
#include <string.h>
struct {
  unsigned int age : 3;
} Age;
int main( ) {
  Age.age = 4;
  printf( "Sizeof( Age ) : %d\n", sizeof(Age) );
  printf( "Age.age : %d\n", Age.age );
  Age.age = 7;
  printf( "Age.age : %d\n", Age.age );
  Age.age = 8;
  printf( "Age.age : %d\n", Age.age );
return 0;
}
```

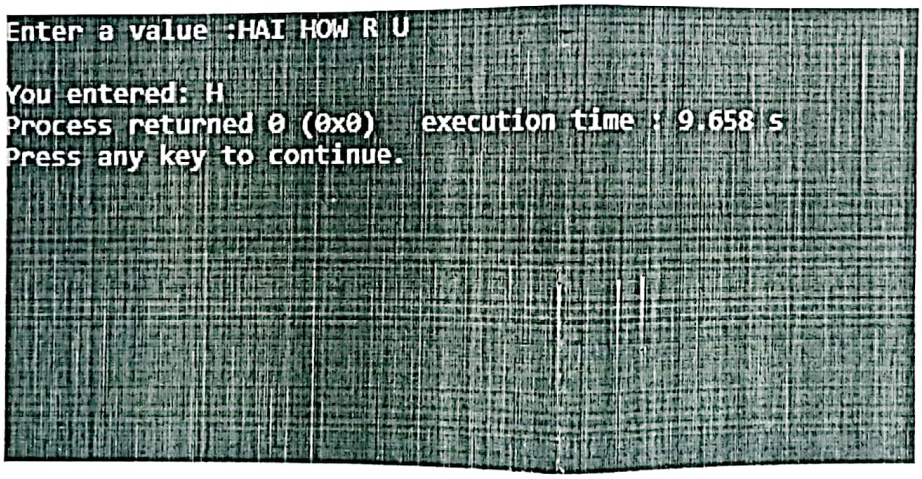
OUTPUT:



C-INPUT & OUTPUT:

```
#include <stdio.h>
int main( ) {
  int c;
  printf( "Enter a value :");
  c = getchar( );
  printf( "\nYou entered: ");
  putchar( c );
 return 0;
}
```

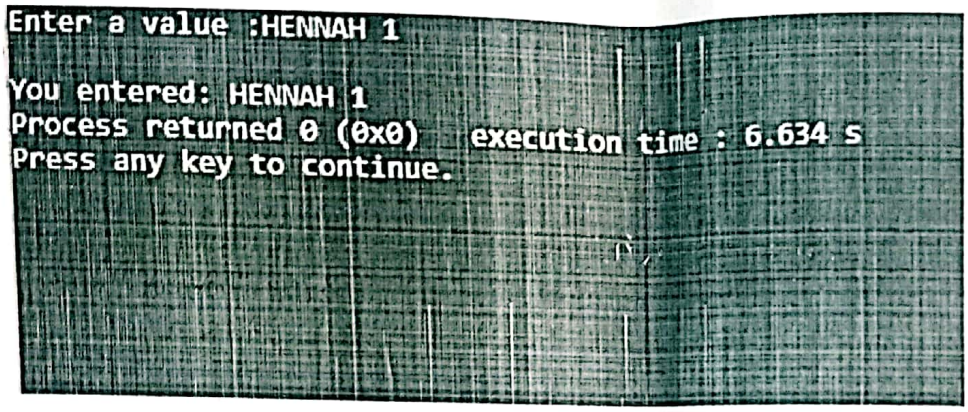
OUTPUT:



SCANF AND PRINTF FUNCTIONS:

```
#include <stdio.h>
int main( ) {

  char str[100];
  int i;
```

```
    printf( "Enter a value :");
    scanf("%s %d", str, &i);

    printf( "\nYou entered: %s %d ", str, i);

    return 0;
}
```

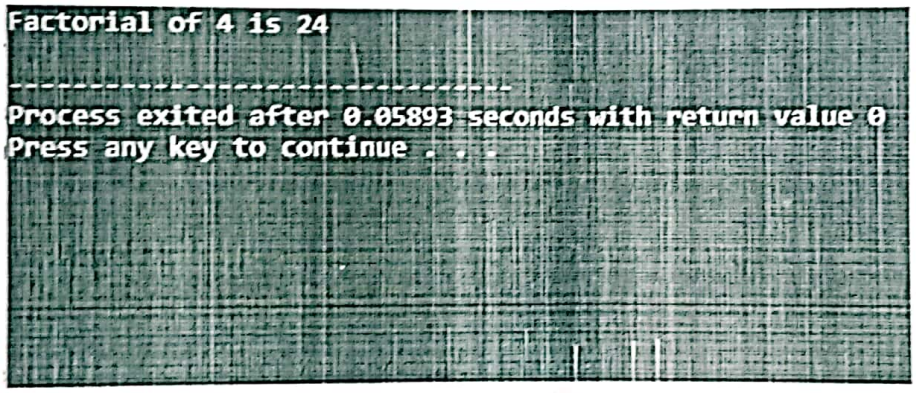


C-RECURRSION:

FACTORIAL:

```
#include <stdio.h>
unsigned long long int factorial(unsigned int i) {
  if(i <= 1) {
    return 1;
  }
return i * factorial(i - 1);
}
int  main() {
  int i = 12;
  printf("Factorial of %d is %d\n", i, factorial(i));
  return 0;
}
```

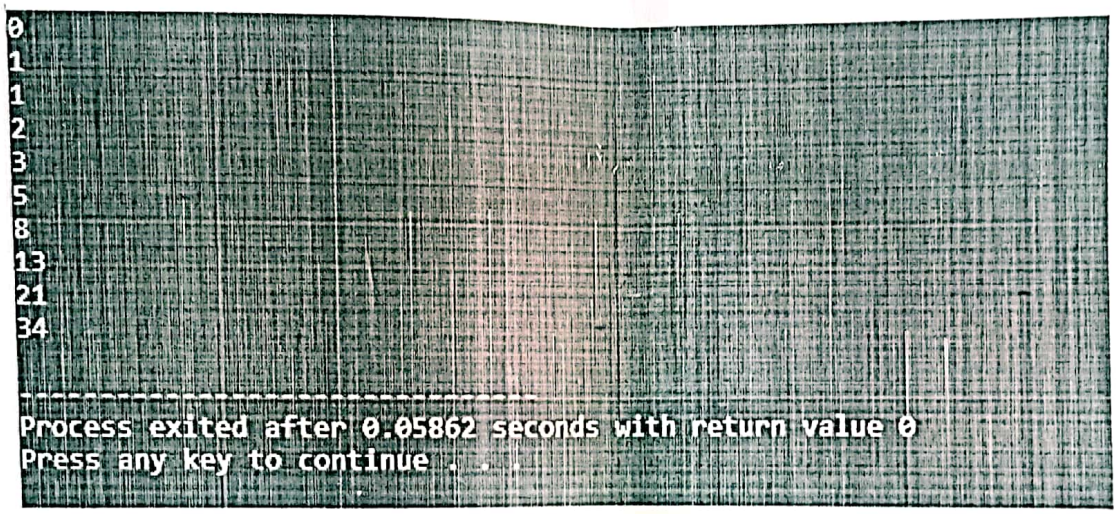
OUTPUT:



FIBONACCI:

```
#include <stdio.h>
int fibonacci(int i) {
  if(i == 0) {
    return 0;
  }
  if(i == 1) {
    return 1;
```

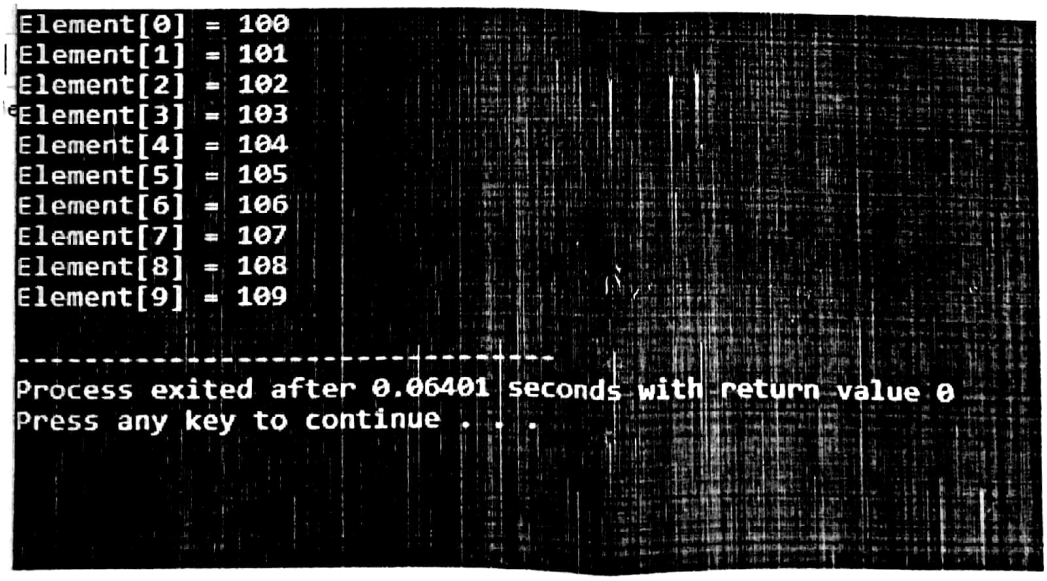
```
  }
  return fibonacci(i-1) + fibonacci(i-2);
}
int main() {
  int i;
  for (i = 0; i < 10; i++) {
    printf("%d\t\n", fibonacci(i));
  }
  return 0;
}
```



C-ARRAYS

```
#include <stdio.h>

int main () {

  int n[ 10 ]; /* n is an array of 10 integers */
  int i,j;

  /* initialize elements of array n to 0 */
  for ( i = 0; i < 10; i++ ) {
    n[ i ] = i + 100; /* set element at location i to i + 100 */
  }

  /* output each array element's value */
  for (j = 0; j < 10; j++ ) {
    printf("Element[%d] = %d\n", j, n[j] );
  }

  return 0;
}
```

OUTPUT:

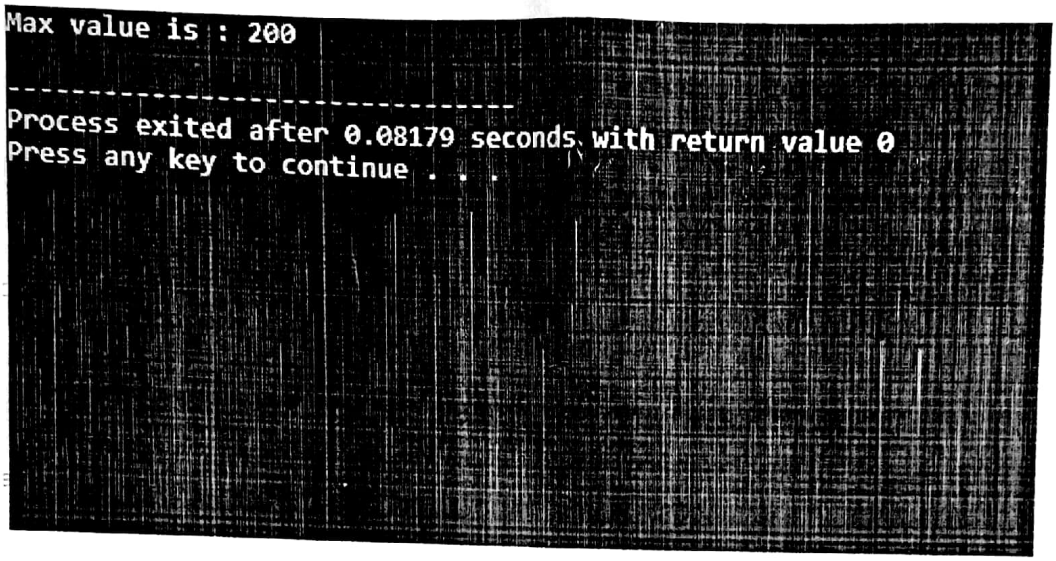


CALLING FUNCTION:

```
#include <stdio.h>

/* function declaration */
int max(int num1, int num2);

int main () {

  /* local variable definition */
  int a = 100;
  int b = 200;
  int ret;

  /* calling a function to get max value */
  ret = max(a, b);

  printf( "Max value is : %d\n", ret );

  return 0;
}

/* function returning the max between two numbers */
int max(int num1, int num2) {

  /* local variable declaration */
  int result;

  if (num1 > num2)
    result = num1;
  else
```

```
    result = num2;

  return result;
}
```

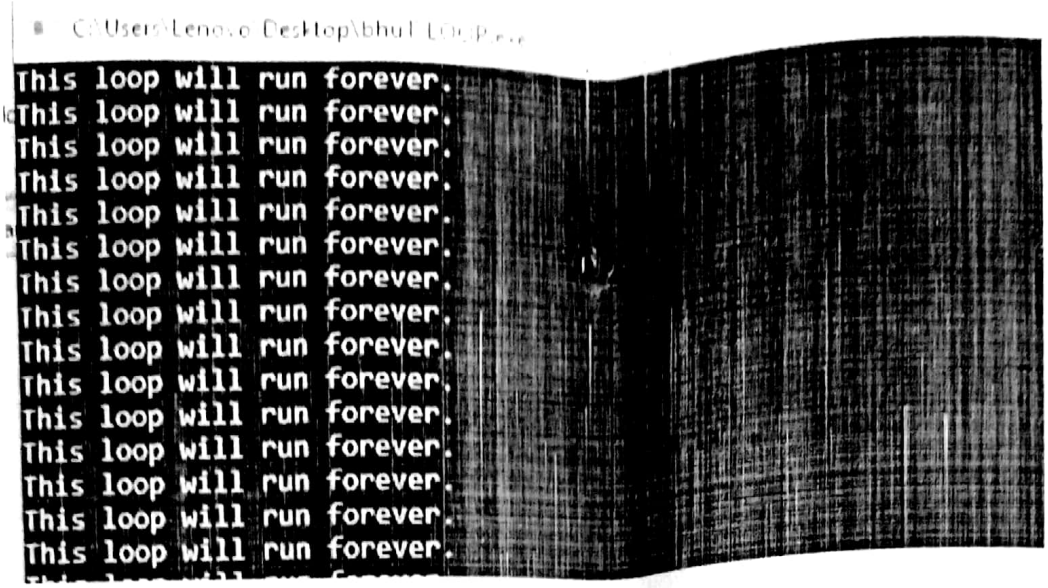
OUTPUT:



C-LOOPS:

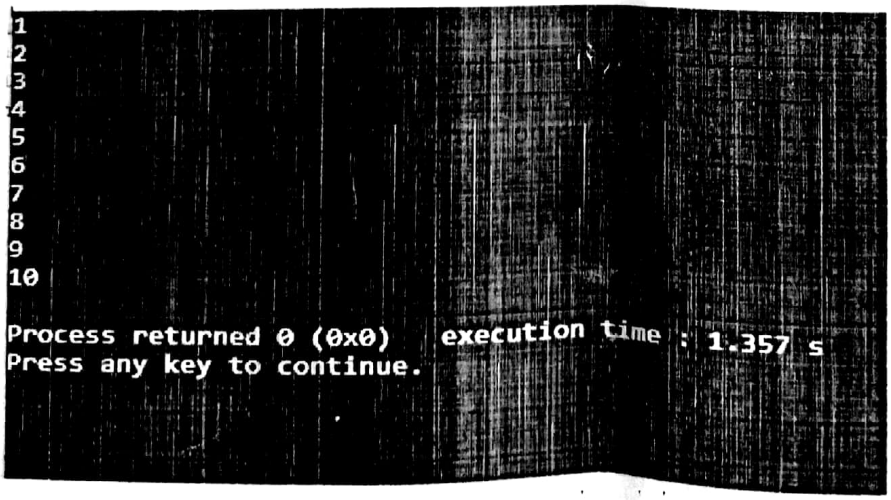
```
#include <stdio.h>

int main () {

  for( ; ; ) {
    printf("This loop will run forever.\n");
  }

  return 0;
}
```

OUTPUT:




```
#include<stdio.h>
#include<conio.h>
int main()
{int num=1;    //initializing the variable
        while(num<=10)       //while loop with condition
        {
                printf("%d\n",num);
                num++;                  //incrementing operation
        }
        return 0;
}
```

OUTPUT:



POINTER-C

```
#include <stdio.h>

int main()

{

int num = 10;

printf("Value of variable num is: %d", num);

/* To print the address of a variable we use %p

format specifier and ampersand (&) sign just

before the variable name like &num.

*/

printf("\nAddress of variable num is: %p", &num);

return 0;

}
```

OUTPUT:

```
Value of variable num is: 10

Address of variable num is: 0x7fff5694dc58
```

POINTER:

```
#include <stdio.h>

int main()

{

  //Variable declaration

  int num = 10;


  //Pointer declaration

  int *p;
```

```
//Assigning address of num to the pointer p
p = #

printf("Address of variable num is: %p", p);
return 0;
}
```

OUTPUT:

```
Address of variable num is: 0x7fff5694dc58
```

VARIABLE ARGUMENTS:

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

double average(int num,...) {

   va_list valist;
   double sum = 0.0;
   int i;

   /* initialize valist for num number of arguments */
   va_start(valist, num);

   /* access all the arguments assigned to valist */
   for (i = 0; i < num; i++) {
     sum += va_arg(valist, int);
   }

   /* clean memory reserved for valist */
   va_end(valist);
```

```
  return sum/num;
}

int main() {
  printf("Average of 2, 3, 4, 5 = %f\n", average(4, 2,3,4,5));
  printf("Average of 5, 10, 15 = %f\n", average(3, 5,10,15));
}
```

OUTPUT:

```
Average of 2, 3, 4, 5 = 3.500000
Average of 5, 10, 15 = 10.000000
```

FILE I/O

```
#include <stdio.h>

main() {

  FILE *fp;
  char buff[255];

  fp = fopen("/tmp/test.txt", "r");
  fscanf(fp, "%s", buff);
  printf("1 : %s\n", buff );

  fgets(buff, 255, (FILE*)fp);
  printf("2: %s\n", buff );

  fgets(buff, 255, (FILE*)fp);
  printf("3: %s\n", buff );
```

```
fclose(fp);


}
```

OUTPUT:

```
1 : This
2: is testing for fprintf...

3: This is testing for fputs..
```