Example Codes that we practiced in Class
You can copy each code and paste in https://ide.geeksforgeeks.org/ that I use in class.

```
//==============
//==============

#include <stdio.h>

int main () {

   int  var1;
   char var2[10];

   printf("Address of var1 variable: %x\n", &var1  );
   printf("Address of var2 variable: %x\n", &(var2[0])  );

   return 0;
}


//==============
//==============
#include <stdio.h>

int main() {
    // declare and initialize a variable
    // declare a pointer
    // store the address of the variable in the pointer

    // print the address of the variable in 2 ways

    // print the value that is stored in the variable in 2 ways

          //code
          return 0;
}

//==============
//==============

#include <stdio.h>

int main () {
```

```
   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer variable*/

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );
   printf("Value of var variable: %d\n", var );
   return 0;
}


//==============
//==============

#include <stdio.h>

int main() {
          //code

    int val = 10;
    int* myPtr = &val;  //declare pointer to val
   // *myPtr = 5;                //dereference pointer and set
    //that location = 5 it means the
    //myPtr content is 5 so any
    //variable at that location is 5.

    printf("val =%d",val);//print val=(value of val)

          return 0;
}


//============
//============

#include <stdio.h>
```

```
int main() {
        //code

int g, grades[ ] = {10, 20, 30, 40 }, myGrade = 100, yourGrade = 85, *pGrades;
/* grades can be (and usually is) used as array name */

for (g = 0; g < 4; g++)
printf("%d\n",grades[g]);

printf("another way \n");

/* grades can be used as a pointer to its array if it doesn't change*/
for (g = 0; g < 4; g++)
printf("%d\n", *(grades + g));
/* but grades can't point anywhere else */

//grades = &myGrade; /* compiler error */
/* pGrades can be an alias for grades and used like an array name */

printf("another way \n");
pGrades = grades; /* or pGrades = &(grades[0]); */
for( g = 0; g < 4; g++)
printf( "%d\n", pGrades[g]);
/* pGrades can be an alias for grades and be used like a pointer that changes */

printf("another way \n");
for (g = 0; g < 4; g++)
printf("%d\n",*(pGrades++));
/* BUT, pGrades can point to something else other than the grades array */
pGrades = &myGrade;
printf( " address location %d\n", &pGrades);
pGrades = &yourGrade;
printf( " address location %d\n", &pGrades);

        return 0;
}
```

==========

Example

==========

```
#include <stdio.h>

int main () {
```

```
 char c, *cptr=&c;
 int i, *iptr=&i;
 double d, *dptr=&d;

  //first print the size of char, int, double
    printf("Char size %d\n", sizeof(char));
    printf("Int size %d\n", sizeof(int));
    printf("Double size %d\n", sizeof(double));

  //next print the address of pointer to c and next one

  //next print the address of pointer to i and next one
  //next print the address of pointer to d and next one
  printf("size of %d %d %d\n", sizeof(char), sizeof(int), sizeof(double) );
    printf("Value of var variable: %p %p %p\n", cptr, iptr,dptr );
    printf("Value of var variable: %p %p %p\n", cptr++, iptr++,dptr++ );
    printf("Value of var variable: %p %p %p\n", cptr, iptr,dptr );


    return 0;
}

=============
=============

#include <stdio.h>

 int main () {
char hello[ ] = "Hello!";
char * ptrChar;
ptrChar = &(hello[3]);
//What is printed from each of the following?
printf("%s\n",hello);
printf("%s\n",ptrChar );
printf("%s\n",&(hello[3]));
printf("%s\n",hello + 3);
printf("%c\n",hello[3]); //x

}
```

==========
Example
==========

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

// define a struct books with members title, author, subject, book_id
//in Main Declare Book1 and Book2 of Struct type Books
//Enter Book1  and Book2 specification one after eachother
// Print Book1 and Book2 specifics

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};



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 All");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Arc");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info */
   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);
```

```
   /* print Book2 info */
   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;
}
```

==========
Example
==========

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

// define a struct books with members title, author, subject, book_id
//Declare a function "printbook" that prints "struct books" specifics.
// In main declare Book1 and Book2 as structs books type
//Enter Book1  and Book2 specification one after eachother
// call the "printbook" function to Print Book1 and Book2 specifics

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 All");
   strcpy( Book1.subject, "C Programming Tutorial");
```

```
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Arc");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info */
   printBook( Book1 );

   /* Print Book2 info */
   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);
}
```

==========
Example
==========

```
#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 Arc");
   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);
}
```

==========
Example
==========

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

union Data {
   int i;
   float f;
   char str[20];
```

```
};

int main( ) {

   union Data data;

   data.i = 10;
   data.f = 220.5;
   strcpy( data.str, "C Programming");
   data.f = 220.5;

   printf( "data.i : %d\n", data.i);
   printf( "data.f : %f\n", data.f);
   printf( "data.str : %s\n", data.str);

   return 0;
}
```

==========
Example
==========

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

union Dataunion {
   int i;
   float f;
   char str[20];
};

struct Datastruct {
   int i;
   float f;
   char str[20];
};

int main( ) {

   union Dataunion data1;
   struct Datastruct data2;

   printf( "Memory size occupied by int : %d\n", sizeof(int));
```

```
   printf( "Memory size occupied by float : %d\n", sizeof(float));
   printf( "Memory size occupied by char : %d\n", sizeof(char));



   printf( "Memory size occupied by data1 : %d\n", sizeof(data1));

   printf( "Memory size occupied by data : %d\n", sizeof(data2));

   return 0;
}
```

==========
Example
==========

```
#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 : 3;
    unsigned int heightValidated : 30;
} 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;
}
```

==========
Example
==========

```
#include <stdio.h>

#include <string.h>

typedef struct point {
   int x,y;
}point_t;

typedef struct line{
    point_t start,end;

} line_t;

/* function declaration */
void printpoint( struct point apoint);
struct point inputpoint();


int main( ) {

 point_t endpoint;
 line_t line1, line2;
// endpoint.x=1;
// endpoint.y=2;
line1.start.x=5;
  endpoint=inputpoint();
   printpoint( endpoint );

   return 0;
}

struct point inputpoint(){
    struct point p;
    printf ("enter x and y coordinates");
    scanf("%d %d\n", &p.x,&p.y);
    return p;
}


void printpoint( struct point apoint){
```

```
   printf( "%2d, %2d\n", apoint.x, apoint.y);
}
```

==========

Example

==========

```
#include <stdio.h>

#include <string.h>

typedef struct point {
   int x,y;
}point_t;

typedef struct line{
    point_t start,end;

} line_t;

/* function declaration */
void printpoint( struct point apoint);
struct point inputpoint();


int main( ) {

 point_t endpoint;
 line_t line1, line2;
 line_t lines[5];
// endpoint.x=1;
// endpoint.y=2;
line1.start.x=5;
line1.start.y=10;

for (int i=0;i<5;i++){
lines[i].start.x=i;
lines[i].start.y=i;
}


printf( "%2d, %2d\n", line1.start.x, line1.start.y);
```

```
for (int i=0;i<5;i++){
    printf( "%2d, %2d\n", lines[i].start.x, lines[i].start.y);
}

   return 0;
}
```

==========
Example
==========

```
#include <stdio.h>


typedef struct month{
 int nrDays;
 char name[3];
}MONTH_t;


int main( ) {

MONTH_t january = {31,"JAN"};

printf( "%c\n", january.name[2]);

   return 0;
}
```

==========
Example
==========

```
#include <stdio.h>

int main () {
   int arr[ 10 ]={5} ;
   int * p1, * p2 ;

    p1 = arr + 1 ;
    p2 = p1 - 1 ;
```

```
 printf("Address of arr: %x\n", arr  );
 printf("size of arr: %d\n", sizeof (arr) );
 printf("Address of p1: %x\n", p1  );
 printf("Address of p2 %x\n", p2 );
 printf("Address of p1: %x\n", & arr[ 1 ]);
 printf("Address of p2: %x\n", & arr[ 0 ]);

 printf("Value of *p1 variable: %d\n", *p1 );
 printf("Value of *p2 variable: %d\n", *p2 );
   return 0;
}
```

==========
Example
==========

```
#include <stdio.h>

int main () {

 char c, *cptr=&c;
 int i, *iptr=&i;
 double d, *dptr=&d;

  //first print the size of char, int, double
    printf("Char size %d\n", sizeof(char));
    printf("Int size %d\n", sizeof(int));
    printf("Double size %d\n", sizeof(double));

  //next print the address of pointer to c and next one

  //next print the address of pointer to i and next one
  //next print the address of pointer to d and next one
  printf("size of %d %d %d\n", sizeof(char), sizeof(int), sizeof(double) );
   printf("Value of var variable: %p %p %p\n", cptr, iptr,dptr );
   printf("Value of var variable: %p %p %p\n", cptr++, iptr++,dptr++ );
   printf("Value of var variable: %p %p %p\n", cptr, iptr,dptr );


   return 0;
```

```
}
```

==========

Example

==========

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {

   char name[100];
   char *description;

   strcpy(name, "Zara Ali");

   /* allocate memory dynamically */
  // description = malloc( 200 * sizeof(char) );
    description = calloc( 200, sizeof(char) );


   if( description == NULL ) {
      fprintf(stderr, "Error - unable to allocate required memory\n");
   } else {
      strcpy( description, "Zara ali a DPS student in class 10th");
   }

   printf("Name = %s\n", name );
   printf("Description: %s\n", description );
   printf("%d\n",sizeof(name));
 //  printf("%d\n",sizeof(description));
}
```

==========

==========

==========

Example

==========

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

```
#include <stdlib.h>
#include <string.h>


typedef struct player
{
 char name[20];
 struct player *teammate; /* can’t use TEAMMATE yet */
} TEAMMATE;

int main () {


TEAMMATE *team, bob, harry, john;
team = &bob; /* first player */
strncpy(bob.name, "bob", 20);
bob.teammate = &harry; /* next teammate */
strncpy(harry.name, "harry", 20);
harry.teammate = &john; /* next teammate */
strncpy(john.name, "bill", 20);
john.teammate = NULL; /* last teammate */

// start with first player
TEAMMATE *t = team; // t is now equal to &bob
// while there are more players...
while (t != NULL) {
 printf("%s\n", t->name); // (*t).name
 // next player
 t = t->teammate; //t=(*t).teammate;
}


   return 0;
}
```

==========
Example
==========

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
```

```
   char name[100];
   char *description;

   strcpy(name, "Zara Ali");

   /* allocate memory dynamically */
   description = malloc( 100 * sizeof(char) );

   if( description == NULL ) {
      fprintf(stderr, "Error - unable to allocate required memory\n");
   } else {
      strcpy( description, "Zara ali a DPS student.");
   }

   /* suppose you want to store bigger description */
   description = realloc( description, 30 * sizeof(char) );

   if( description == NULL ) {
      fprintf(stderr, "Error - unable to allocate required memory\n");
   } else {
      strcat( description, "She is in class 10th");
   }

   printf("Name = %s\n", name );
   printf("Description: %s\n", description );

   /* release memory using free() function */
   free(description);
}
```

==========
Example
==========

```
#include <stdio.h>

#include <stdio.h>
typedef double Radius;
//#define PI 3.141
/* given the radius, calculates the area of a circle */
```

```
double circleArea( double radius ){
 return ( 3 * radius * radius );
}

// given the radius, calculates the circumference of a circle
double circumference( Radius radius ){
 return (2 * 3 * radius );
}


int main( ){
 double radius = 4.5;

 double area = circleArea( radius );

 return 0;
}
```

Example 17

```
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;
}
```

======

Example 18

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

```
int main () {

   char str1[12] = "Hello"
   char str2[12] = "World";
   char str3[12];
   int  len ;

   len = strlen(str1);
   printf("strlen(str1) :  %d\n", len );

   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) :  %s\n", str3 );

   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );

   /* total lenghth of str1 after concatenation */
   len = strlen(str1);
   printf("strlen(str1) :  %d\n", len );

   return 0;
}

//==============
//==============
#include <stdio.h>

int main () {

   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
   printf("Greeting message: %s\n", greeting );
   return 0;
}

//==============
//==============


#include <stdio.h>
```

```
int ArraySum(int array[], int size){
int k, sum=0;
for (k=0; k<size; k++)
 sum+= array[k];
return sum;
}

int main(){
 int ages[6] = {19,18,17,22,44,66};
 int sumAge = ArraySum(ages,6);
 printf("The sum of ages is %d\n",sumAge);
 return 0;
}


//=============
//=============


#include <stdio.h>

 int main ( )
{
    /* a double, a pointer to double,
    ** and a pointer to a pointer to a double */
    double gpa = 3.25, *pGpa, **ppGpa;
    /* make pgpa point to the gpa */
    pGpa = &gpa;
    /* make ppGpa point to pGpa (which points to gpa) */
    ppGpa = &pGpa;
    // what is the output from this printf statement?
    printf( "%0.2f, %0.2f, %0.2f", gpa, *pGpa, **ppGpa);
    return 0;
}

//=============
//=============

#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;

struct {
   unsigned int widthValidated : 1;
    unsigned int heightValidated : 30;
} status3;

struct {
   char widthValidated : 3;
    char heightValidated : 2;
} status4;


int main( ) {

    printf( "Memory size occupied by char : %d\n", sizeof(char));
   printf( "Memory size occupied by  float : %d\n", sizeof(float));
   printf( "Memory size occupied by status1 : %d\n", sizeof(status1));

//=============
//=============


double getAverage(int arr[], int size);


int main () {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17, 50};
   double avg;

   /* pass pointer to the array as an argument */
   avg = getAverage( balance, 5 ) ;
```

```
   /* output the returned value  */
   printf("Average value is: %f\n", avg );
   return 0;
}

double getAverage(int arr[], int size) {

   int  i, sum = 0;
   double avg;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

//===========
//===========

#include <stdio.h>

 int main ( )
{
    /* a double, a pointer to double,
    ** and a pointer to a pointer to a double */
    double gpa = 3.25, *pGpa, **ppGpa, ***pppGpa;
    /* make pgpa point to the gpa */
    pGpa = &gpa;
    /* make ppGpa point to pGpa (which points to gpa) */
    ppGpa = &pGpa;

    pppGpa = &ppGpa;
    // what is the output from this printf statement?
    printf( "%0.2f, %0.2f, %0.2f, %0.2f", gpa, *pGpa, **ppGpa, ***pppGpa);
    return 0;
}

//=============
//=============
```