STRUCT & UNION

STRUCT DATA TYPE

Simple abstract data type	**struct** Color { int r; int g; int b; }; // sizeof(struct Color) == 12
	
	struct Color test;
	test.h = 255; test.w = 255; test.d=255;
	// or struct Color test = { 255, 255, 255 };
	
	OR
	
	typedef struct { int r; int g; int g; } Color;
	
	Color test;
	test.h = 255; test.w = 255; test.d=255;
	// or Color test = { 255, 255, 255 };

	typedef struct { int base; int height; int depth; Color c ; } Box; Box Test; Test.base = 50; Test.height = 3; Test.depth = 5; Test.c.r = 255, Test.c.g=255; Test.c.b = 0; or ultimately you can also initialize it at the time when it is declared: Box Test = { 50, 3, 5, {255,255,0} };

Structure Array	typedef struct {
	int time;
	float velocity;
	float mass;
	} Object;
	
	void main()
	{
	Object bots[2] = { { 50, 9.5, 10.5 }, { 20, 20, 5.5 } };
	for(int i = 0; i< 2; i++)
	cout << "object-" << i << ": " << bots.velocity / obs.time)* bots.mass << endl;
	}

UNION DATA TYPE

```
typedef union {
   float fVal;
   int iVal;
} MyUnionType;
```

sizeof(myUnionType) == 4 , not 8

Try the following:

```
typedef unsigned char ubyte;
typedef struct { ubyte r; ubyte g; ubyte b; } Color;

typedef union {
       int code;
       Color c;
} ComboType;

ComboType combo;

… in a function, do this :

       combo.c.r = 255; // or 0xff
       combo.c.g = 255; // of 0xff
       combo.c.b = 0;

       printf("%d", combo.code);
```

Try the following:

a) Create a short program to create the union type and initialize the r,g,b only.
b) Build and run in debugger. Stop right after you initialize the color.
c) Then, also look at the "Watch tab" in the output window.
d) Put the parameter and check out the change in the code field. You should experiment by entering different values.

In hex (base-16) display

Watch 1

Name	Value
combo.c.r	0x10 '\x10'
combo.c.g	0x08 '\b'
combo.c.b	0xff 'ÿ'
combo.code	0x00ff0810

or

In Dec (base-10) display

Watch 1

Name	Value
combo.c.r	10 '\n'
combo.c.g	8 '\b'
combo.c.b	255 'ÿ'
combo.code	16713738

Watch 1

Name	Value
combo.c.r	0xff 'ÿ'
combo.c.g	0x0a '\n'
combo.c.b	0x08 '\b'
combo.code	0x00080aff

or

Watch 1

Name	Value
combo.c.r	255 'ÿ'
combo.c.g	10 '\n'
combo.c.b	8 '\b'
combo.code	527103

ADVANCED TOPICS - DATA ALIGNMENT

In college, this may fall in computer data architecture or compiler course. Different machine architecture does it slightly different. In order to help the CPU fetch data from memory in an efficient manner, data is being arranged in N-bytes chunk, mostly 4-bytes. This is called data alignment.

Every data type has an alignment associated with it which is mandated by the processor architecture rather than the language itself. Aligning data elements improves performance as it allows the processor to fetch data from memory more efficiently.

Examples:

1	struct TILE {
	TILE *right;
	int id;
	char s[6];
	char a;
	};
		sizeof (TILE) == 16 , not 15
		0 th
		4 th
		8 th
		14 th

2	struct TILE {
	char a;
	TILE *right;
	int id;
	char s[6];
	char b;
	};
		sizeof (TILE) == 20, not 16
		0 th
		4 th
		8 th
		12 th
		18 th

3	struct TILE {
	short a;
	TILE *right;
	int id;
	char s[6];
	char b;
	};
		sizeof (TILE) == 20, not 17
		0 th
		4 th
		8 th
		12 th
		18 th
		But note: sizeof(short) == 2

4	struct TILE {
	short a;
	TILE *right;
	int id;
	char s[7];
	char b;
	char c;
	};
		sizeof (TILE) == 24, not 19
		0 th
		4 th
		8 th
		12 th
		19 th
		20 th

BITS FIELD IN C STRUCTURE

You can create variables which represent a bit!
(note: ubyte is a user-defined type : typedef unsigned char ubyte)

```
typedef struct {
        ubyte N : 1;
        ubyte NE : 1;
        ubyte E : 1;
        ubyte SE : 1;
        ubyte S : 1;
        ubyte SW : 1;
        ubyte W : 1;
        ubyte NW : 1;
} BitsPACKET;
```

sizeof(BitsPACKET) == 1 byte, not 8 bytes.

WATCH OUT THE BITS ORDER:

```
typedef union  {
        BitsPACKET bits;
        ubyte num;
} unDir;

void main()
{
        unDir dir;
        memset(&dir, 0, sizeof(dir));

        dir.bits.N = 1;
        dir.bits.W = 1;

        // 1  0  0 0 0 0  1 0
        // N  NE E E S SW W NW

        // being stored as  0x41
        // 0  1  0  0  0  0  0   1
        // NW W SW  S  E  E  NE  N

        printf("%d", dir.num);
}
```

RECALLING USING ENUMERATION TYPE

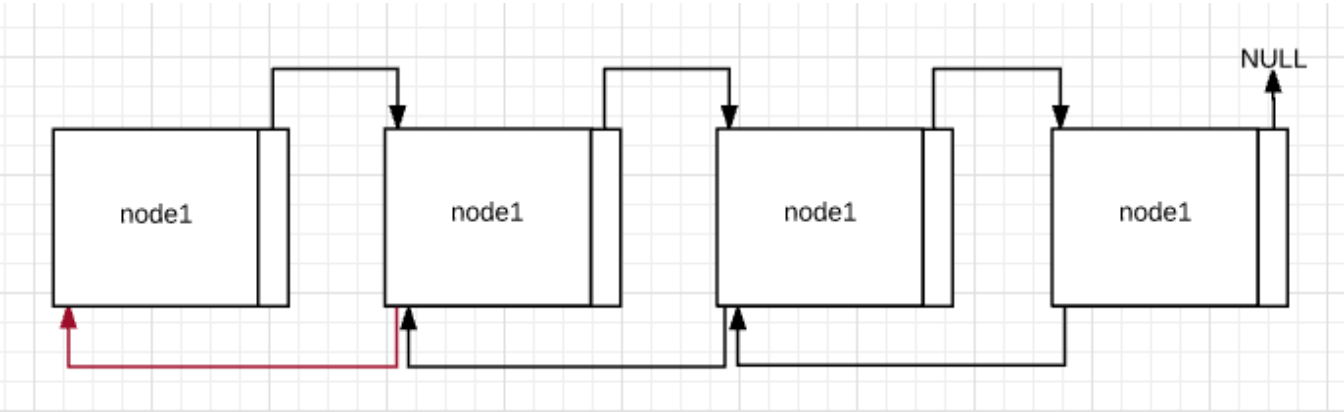
Take the following macros:	Use Enumeration instead:
#define NORTH 0 #define SOUTH 1 #define EAST 2 #define WEST 3 e.g. int wind_direction = NO;	enum Directions { NORTH, SOUTH, EAST, WEST}; Directions dir; Valid : dir = NORTH; dir = WEST ; dir = (Directions)3; Bad : dir = (Directions) 4;
use it with array	e.g. : char directions[WEST]; sizeof(directions) is 3 char directions[WEST+1]; sizeof(directions) is 4 char directions[][10] = { "NORTH", "EAST", "SOUTH", "WEST" }; sizeof(directions) == 40

DEMONSTRATE FURTHER USAGE:

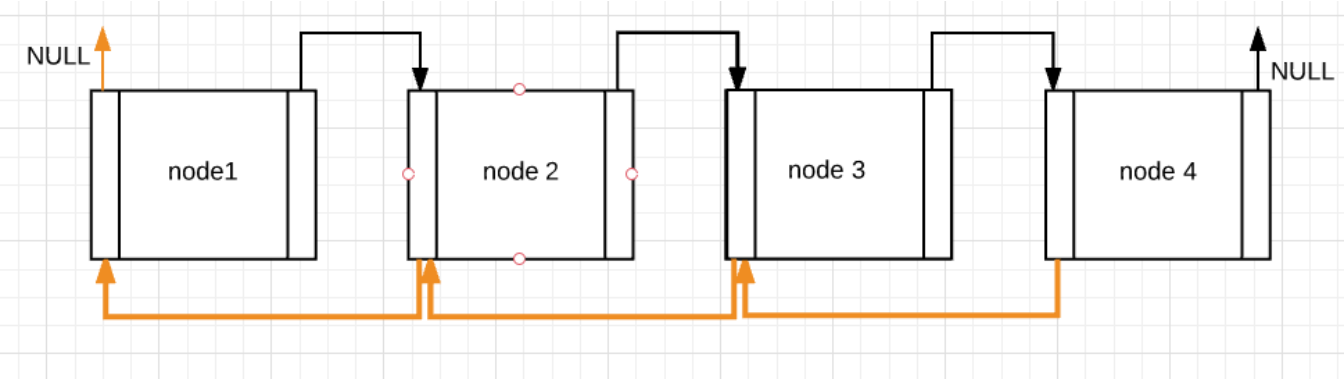
```
enum enDir { N, NE, E, SE, S, SW, W, NW, Last};

char sDir[][10] = { "N", "NE", "E", "SE", "S", "SW", "W", "NW", "   " };

enDir d;

for ( d = N;    d < Last;     d = (enDir)(d+1) )              // ie. From 0 to 7
{
        if ( checkDir(dir, d) )
                printf("%s  is set\n", sDir[d]);
        else
                printf("%s  is not set\n", sDir[d]);
}
```

Linked List

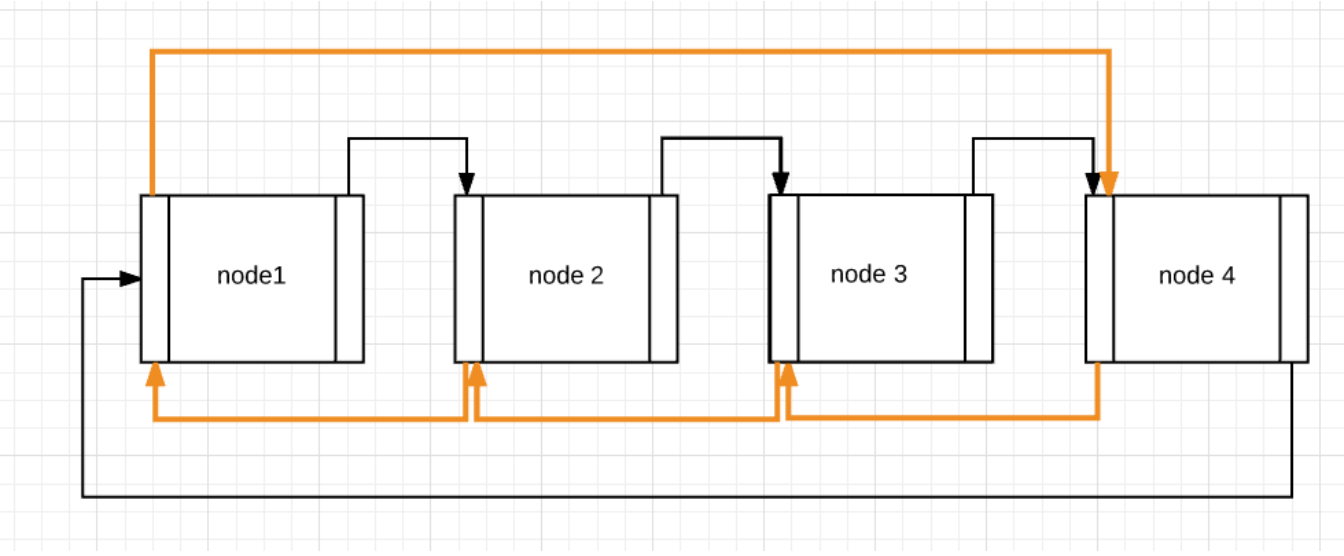
Single linked list



Double linked list



Circular linked list



Accessing time of an element :	O(n)
Search time of an element :	O(n)
Insertion of an Element :	O(n)
Deletion of an Element :	O(n)

EXERCISES :

1) Take social security number : ###-##-####
   - 1st 3 numbers : which State
   - 2nd 2 numbers : group
   - Last 4 numbers : serial number

   Write a program using Union structure to allow user to enter a full social security number such as "111223456". Then, you serial number without additional expression.

   Input Display:
   Enter your SSN (##########) : **111223456**

   Output:
   Region: 111
   Group : 22
   Serial Number : 3456

2) Color Code usually presented in hex, $RRGGBB_{16}$.

   For example: $66CCFF_{16}$, i.e. R== 66_{16}, G=CC_{16}, B=FF_{16}, that gives

   Write a function to ask user to input the value of Red, Green, and Blue. Then, it should produce the final color value in Hexadecimals.

   Sample function prototype:

   ```
   int createColor( unsigned char red, unsigned char green, unsigned char blue) ;
   ```

   // create your own data type ***ubyte*** instead of **unsigned char**

   Your console output should look like this: (red bold font indicates user input)

   ```
   Enter R: (0 <=x <= 255) : 102
   Enter G: (0 <=x <= 255) : 204
   Enter B: (0 <=x <= 255) : 255

   Output Color : 66ccff
   ```

After that, go to http://www.w3schools.com/colors/colors_picker.asp to validate the color. You should see this color:

Tips for forming the coder code: use bits shift , such as R<<16 | …

3) Create a function to take in a RGB color code, such as FFFF00, then it should generate the individual R. G, B value. E.g.

Sample function prototype: void makeColor(unsigned int rgb, ubyte *red, ubyte *green, ubyte *blue) ;

Your console output should look like this:

```
Enter the RGB Code : 66ccff

Your output should look like:
R (in hex) = 66 or 102
G (in hex) = cc or 204
     B (in hex) = ff or 255
```

4) Modify (3) to populate a Color Structure Type field instead of 3 separate fields:

Sample color structure type:

```
typedef struct {
int red;
int green ;
int blue;
}Color;
```

Your function prototype:

void makeColor(int, Color *);

// 1st parameter is the color code such as 0xff00ff.

Go the web page again to validate your conversion.

if you use C++ standard IO : use the "hex" manipulator. E.g. cout << hex << theNumber
if you use C standard IO : use the "0x" manipulator. E.g. printf("%x", theNumber);

5) Refer to the Bits Fields in Structure sample above. You will write two functions:

```
void setBits(unDir *, enDir)
bool getBits(unDir, enDir)
```

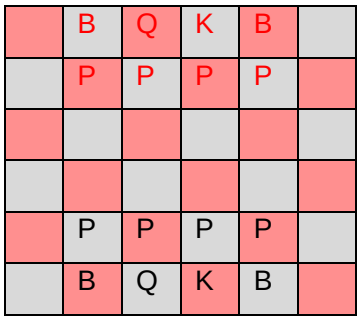
If you do not know pointer, just make the unDir field as a global instead. Assuming a global field unDir Direction already exists. Your setBits(…) function will set the bit field inside this structure. Thus, your 1st function prototype will be :

```
void setBits(enDir)
bool getBits(enDir)
```

6) Write a program to allow two users to simulate a partial Chess game with only 2 colored (red, black) groups of 8 pieces/group:

1 Queen, 1King, and 2 Bishops, 4 Pawns

Startup:



As you see this is not a complete 8x8 chess board but only 6x6, nor does it have all pieces. The goal of this program is not to do AI Chess game, but to exercise proper design with struct, enum, functions, and possibly union if applicable.

You do not need to be a proficient chess player, but need to know the rules of a kind, queen and bishop, and pawn.

Legal moves:

- The bishop: only in a straight line diagonally for any number of squares
- The queen: in any number of squares in a straight line horizontally, vertically or diagonally.
- The king: in any direction, including diagonally, but it can only move one square at a time.
- The pawn:
  - Move one square forward only, not backward.
  - If it is the pawn's first move, it can move one or two squares directly forward; but only one at a time after 1st move.

To capture:

- The bishop, queen and king:
  - capture others in their legal path, except when there is another piece (like its own color) in the way.
- The pawn:
  - cannot capture a piece directly in front of it.
  - Capture a piece only by moving one square forward diagonally.