

CODE CONVENTIONS

Copyright© CS 101 Team 232 (2011-2012) [Created by Guna Prasaad on 05th Oct 2011]

Development Environment:

It is recommended to use gedit software to develop the source codes.

If you are using windows you can download gedit from internet or can get it from me.

Font :

We will follow a uniform font 'Monospace Normal' at font size = 12.

Naming of files:

The name of the file has to be a mnemonic of the objective of the source code written. For example: the file for range update of pawn should be as follows:

`pawn_range_update.cpp`

When you are asked to write function for a particular objective: Do not include the main () function in it. Just write the function definition.

Beginning Comments:

All C++ files that are submitted for testing and debugging should contain a beginning comment which must necessarily address the following Queries.

```
/*Brief Description of the objective of this file
```

```
*@version 1.0 <date> eg: 19 Oct 2011
```

```
*@author FirstName LastName
```

```
*/
```

Inclusion of header files:

Include all the header files needed and DONOT forget to initialize the namespace.

Prototype of functions:

The prototype of all the functions written in the particular cpp file must be written at the top before defining any of the functions.

Indentation Levels:

Normal indentation levels as per logic. Tab width must be set at 4.

Implementation of Comment Formats:

ITERATIONS : Each iteration should be preceded by a block comment explaining the algorithm and test condition

Block Comment format:

```
/*  
*This is an example  
*to illustrate to you  
*how to put a block comment  
*/
```

NOTE :

1. Except the first and last line all lines must be preceded by an asterisk '*'
2. The comment must also be indented the same way as the scope. For example if you are going to write a block comment for a for loop then the for loop and comment must be at the same indentation.
3. The block comment should be preceded and succeeded by a blank line.

CONDITIONAL STATEMENTS: As the first statement under each conditional statement there must be a single line comment briefly explaining the condition and the corresponding execution algorithmically.

Similarly they must be indented according to the scope.

Example:

```

If(x>=0&&x<=7&&y<=7&&y>=0){
    /* check if point lies in board to include in range */
    Nrange++;
    Range[Nrange].copypos(pos[x][y]);
}

```

DECLARATIONS:

1. Declaration of multiple variables in the same line must be avoided. Declare each variable in separate lines as it encourages commenting.
2. Declarations must be as follows:

```
<data type><tab><variable>;
```

NAMING CONVENTIONS:

Identifier Type	Naming Conventions	Examples
Classes	Class names should be <u>nouns</u> , in mixed case with the <u>first letter</u> of each <u>internal word</u> capitalized. Try to keep your class names simple and descriptive. Use whole words—avoid acronyms and abbreviations	class Coin class Movement class Position
Methods or functions	Methods should be <u>verbs</u> , in mixed case with the <u>first letter</u> lowercase, with the <u>first letter</u> of <u>each internal word</u> capitalized.	updateRange(); setX(); setY(); replaceCoin();
Variables	Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic— that is, designed to indicate to the casual observer the intent of its use. One-character	Int i; Char *cp; Float myWidth;

	variable names should be avoided except for temporary “throwaway” variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.	
Macros	For defining macros, all the letters must be capitalized	#define PAWN 5
Temp objects	When declaring temporary objects of a class always prefix ‘temp’ before the name of the variable.	Employee tempName;

APPEARANCE OF STATEMENTS:

1. Simple statements: Each line should contain atmost a single statement. More than one statement in a single line is strictly prohibited. Proper indentation is a must.

2. If..elseif..else statements:

```
if<space>(condition1)<space>{
    ...
}<space>elseif<space>(condition2)<space>{
    ...
}<space>else<space>{
    ...
}
```

Note: Be careful about the white spaces. It must be as in the above format.

3. For statements:

```
for<space>(initialization;<space>condition;<space>updation){
    Statements;
}
```

4. while Loop:

```
while<space>(condition)<space>{
    Statements;
}
```

5. Do while Loop:

```
do<space>{
    Statements;
}<space>while<space>(condition);
```

6. Switch statements:

```
switch<space>(condition)<space>{
case<space>ABC:
```

```

        statements;
        break;
case<space>DEF:
    statements;
    break;
default:
    statements;
    break;
}

```

BLANK LINES:

1. Double blank lines must be left between class and method definitions
2. Single lines must be left in following circumstances:
 - a. Between methods or function definitions
 - b. Between local variables in a method and its first statement
 - c. Between block comments(refer BLOCK COMMENTS)
 - d. Between logical sections in a method to enhance readability

BLANK SPACES

1. A keyword followed by a parenthesis must be separated by a blank space.
For example: while<space>(condition)<space>{
2. A blank space should appear in commas after arguments
3. All binary operations must be separated by spaces. Unary operators such as ++ or – need not be. A+B must be written as A + B. and (a+b)/(c*d) must be written as
=> (a + b) / (c * d)