2020 年度　工学システム学類
プログラミング序論 B　【レポート課題２】　※演習の課題は別にあります。

学籍番号: Student ID（202018021）　氏名:Name（Fu, Ziyu）

以下の設問 1)から 3)に回答せよ。解答スペースは自由に変更して構わない。

---

1) 間接演算子と乗算演算子の使い分け方を説明せよ。

2) ポインタ変数 ptr にアドレス情報が格納されている場合、「*ptr」によって示される情報は何か。

3) 3x3 配列 mat[3][3]に対して
(ア)mat によって示される情報は何か。
(イ)次の(A)と(B)を埋めよ。
*(mat+(3*2+1)) は mat[(A)][(B)]と同じ

---

1) Explain how to use the “indirect operator” and “multiplication operators”.
2) When an address information is stored at the pointer variable “ptr", what is the information indicated by "*ptr”?
3) For a 3x3 array mat[3][3],
   a) Answer what information is presented by the “mat”.
   b) Fill in the following (x) and (y).
   *(mat+(3*2+1)) is equal to mat[(x)][(y)].

(Reponses are on the next page.)

1) The indirect operator is an operator used to obtain the value of a variable through a pointer. It is denoted by an asterisk (*****). The operand of an indirect operator must be a pointer (address).
   The multiplication operator is also denoted by an asterisk (*****). However, it carries out the algebraic multiplication between any two numerical types.
2) **`*ptr`** gives the value of the variable that is stored at the address **`ptr`**.
3)
   a) **`mat`** gives the address of the 0th element of the array, which is also the base address of the matrix.
   b) This is an interesting problem. According to the explanations given on the lecture materials (slides), this problem should be solved like this:
      `*(mat + (3*2 + 1))` = **`*(mat + 7)`**
      = the value in **`mat`** at position 7
      = **`mat[2][1]`**
      `x = 2, y = 1`
      Since each row of the 2D array has 3 columns (or elements), adding **`3*2`** to the pointer value offsets 2 rows in the 2D array, and adding an extra 1 would give the value at position **`[2][1]`** on the 2D array. This explanation does make logical sense.

      However, the notation didn't work for me in the context of 2D pointer manipulations. A 2D array is an array of arrays. If I declare a 2D array **`A[X][Y]`**, each element of a is an array of length **`Y`** by itself.

      In this case, **`*(A + N)`** would therefore return an **`int*`**, which would be the base address of the N-th array in **`A`**, which would be **`N*Y*(sizeof(int))`** away from the base address. Hence when I actually run the code, **`*(mat+7)`** returned an address value **`7*3*4 = 84`** away from the base address.

      Therefore I believe in order to get to **`mat[2][1]`**, the correct expression should be **`*(*(mat+2)+1)`**. It first finds the base address for the 2nd row by **`*(mat+2)`**, and then displace that address by **`1`**, and taking the value of that address would give you the value of **`mat[2][1]`**.