```
*****************************************************************************************
* Name: Cha Yoonsung
* Student Id: 20163162
* Program Id: Hw 1-1
* Description: 파일로부터 희소행렬을 입력받아 전치행렬을 만들고 출력하는 프로그램
* Algorithm: 파일로부터 입력 받은 데이터를 구조체 배열(벡터로 구현) 각 인덱스 별로 row, col, value에
*             저장한 후 입력받은 희소행렬을 fastTranspose algorithm을 이용하여 전치행렬로 변환시킨다.
* Variables
*   - row: 각 구조체의 행 번호          - col: 각 구조체의 열 번호
*   - tRow: 총 행의 개수                - tCol: 총 열의 개수
*   - tValue: 총 원소의 개수
*   - input: 입력 받은 희소행렬 데이터를 저장하는 구조체 벡터
*   - output: 희소행렬을 전치시킨 행렬을 저장하는 구조체 벡터
* Function
*   - fastTranspose() : 희소행렬을 전치시키는 함수
*****************************************************************************************
#include <fstream>
#include <iostream>
#include <vector>

using namespace std;

struct Term {
  int row;
  int col;
  int value;
  Term(int r, int c, int v) : row(r), col(c), value(v) {}
};

vector<Term> fastTranspose(vector<Term> a, int tCol, int tValue);

int main() {
  vector<Term> input;
  vector<Term> output;
  int tRow, tCol, tValue;

  ifstream infile;
  infile.open("HW1-1.dat", ios::in);

  if (infile.fail()) {
    cout << "Cannot open the input file" << endl;
    exit(1);
  }

  infile >> tRow >> tCol >> tValue;

  while (!infile.eof()) {
    int r, c, v;
    infile >> r >> c >> v;
    Term temp(r, c, v);
    input.push_back(temp);
  }

  infile.close();

  cout << "====== Sparse Matrix ======" << endl;
  cout << "\trow\tcol\tvalue" << endl;
  cout << "total\t" << tRow << "\t" << tCol << "\t" << tValue << endl;

  for (int i = 0; i < input.size(); i++) {
    cout << "\t" << input[i].row << "\t" << input[i].col << "\t" << input[i].value << endl;
  }
```

```
  output = fastTranspose(input, tCol, tValue);

  cout << "====== Transpose Matrix ======" << endl;
  cout << "\trow\tcol\tvalue" << endl;
  cout << "total\t" << tCol << "\t" << tRow << "\t" << tValue << endl;

  for (int i = 0; i < output.size(); i++) {
    cout << "\t" << output[i].row << "\t" << output[i].col << "\t" << output[i].value << endl;
  }

  return 0;
}

*****************************************************************************************
* Function: fastTranspose(vector<Term> a, int tCol, int tValue)
* description
*   - 희소행렬과 총 열의 개수, 총 원소의 개수를 parameter로 넘겨받아
*     희소행렬의 행과 열을 전치시켜 반환하는 함수.
*   - 전치행렬을 초기화 한 후 희소행렬 value의 개수가 1개 이상인지 검사 후 진행
*   - 첫 번째 For문
*     - countCol 초기화
*   - 두 번째 For문
*     - 희소행렬 각 열 번호 별 개수를 저장
*   - 세 번째 For문
*     - startingPos에 전치행렬의 row별 시작 위치 저장
*      ex) starting[0] = countCol[0]++;
*     - 해당 startingPos에 접근 시 마다 +1 증가
*   - 네 번째 For문
*     - 전치행렬 b의 시작 위치를 b[startingPos[i]]로 정하고 희소행렬의 row, col, value값 저장
*      ex) b[startingPos[i]].row = a[i].col
*          b[startingPos[i]].col = a[i].row
*          b[startingPos[i]].value = a[i].value
*****************************************************************************************
vector<Term> fastTranspose(vector<Term> a, int tCol, int tValue) {
  vector<Term> b;
  for (int i = 0; i < a.size(); i++) {
    Term temp(0, 0, 0);
    b.push_back(temp);
  }

  int countCol[tCol], startingPos[tCol];
  int j;

  if (tValue > 0) {
    for (int i = 0; i < tCol; i++) countCol[i] = 0;
    for (int i = 0; i < tValue; i++) countCol[a[i].col]++;

    startingPos[0] = 0;
    for (int i = 1; i < tCol; i++) startingPos[i] = startingPos[i - 1] + countCol[i - 1];

    for (int i = 0; i < tValue; i++) {
      j = startingPos[a[i].col]++;
      b[j].row = a[i].col;
      b[j].col = a[i].row;
      b[j].value = a[i].value;
    }
  }

  return b;
}
```

```
*****************************************************************************************
* Name: Cha Yoonsung
* Student Id: 20163162
* Program Id: Hw 1-2
* Description: 도형의 이름과 변의 길이 or 반지름을 데이터로 가지는 파일을 입력 받아 각 도형의 넓이를
*              구하는 프로그램
* Algorithm
*    - Data를 불러와 circle일 경우 2, triangle인 경우 1, rectangle인 경우 0을 반환하도록 하여
*      각 case별로 변의 길이 or 반지름을 구조체 변수에 대입한 후 area 산출
* Variables
*    - fig: 구조체 벡터
*    - name: 도형의 이름              - radius: 원의 반지름
*    - aSide, bSide: 두 변의 길이   - area: 도형의 넓이
*****************************************************************************************
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>

using namespace std;

struct Figure {
  string name;
  double aSide;
  double bSide;
  double radius;
  double area;
  Figure(string n, double a, double b) : name(n), aSide(a), bSide(b) {}
  Figure(string n, double r) : name(n), radius(r) {}
};

int main() {
  vector<Figure> fig;

  ifstream infile;
  infile.open("HW1-2.dat", ios::in);

  if (infile.fail()) {
    cout << "Cannot open the input file" << endl;
    exit(1);
  }

  cout << fixed;
  cout.precision(2);

  while (!infile.eof()) {
    string name;
    infile >> name;
    switch (name == "circle" ? 2 : (name == "triangle" ? 1 : 0)) {
      case 2: {
        double radius;
        infile >> radius;
        Figure temp(name, radius);
        temp.area = pow(temp.radius, 2) * 3.14;
        fig.push_back(temp);
        break;
      }
```

```
        case 1: {
          double a, b;
          infile >> a >> b;
          Figure temp(name, a, b);
          temp.area = (temp.aSide * temp.bSide) / 2;
          fig.push_back(temp);
          break;
        }
        case 0: {
          double a, b;
          infile >> a >> b;
          Figure temp(name, a, b);
          temp.area = temp.aSide * temp.bSide;
          fig.push_back(temp);
          break;
        }
      }
    }

    infile.close();

    for (int i = 0; i < fig.size(); i++) {
      if (fig[i].name == "circle")
        cout << fig[i].name << " " << fig[i].radius << "\t\t" << fig[i].area << endl;
      else
        cout << fig[i].name << " " << fig[i].aSide << " " << fig[i].bSide << "\t" << fig[i].area << endl;
    }

    return 0;
}
```