```
#include "rhino.h"

Rhino::Rhino()
{
        //cout << "Calling Default constructor ... " << endl;
        r = new Rhino[];
        nickname = " ";
        year = 2013;
        children = 0;
        mother = NULL;
        father = NULL;
}
Rhino::Rhino(string n, int y, int m, char g)
{
        //cout << "Calling Nondefault constructor ... " << endl;
        r = new Rhino[];
        nickname = n;
        year = y;
        month = m;
        gender = g;
        children = 1;
        tag = -1;
        mother = NULL;
        father = NULL;
}
Rhino::Rhino(string n, int y, int m, char g, int t, Rhino * mom, Rhino * dad)
{
        //cout << "Calling Second Nondefault constructor ... " << endl;
        r = new Rhino[];
        nickname = n;
        year = y;
        month = m;
        gender = g;
        tag = t;
        mother = mom;
        father = dad;
}
Rhino::~Rhino()
{
        //cout << "Calling destructor ..." << endl;
}
//      -----Set/Get name----- *working*
void Rhino::setName(string nm)
{
        nickname = nm;
};
string Rhino::getName()
{
```

```
        return nickname;
};
//      -----Set/Get year----- *working*
void Rhino::setBirthYear(int y)
{
        year = y;
};
int Rhino::getYear()
{
        return year;
};
//      -----Set/Get month----- *working*
void Rhino::setBirthMonth(int m)
{
        month = m;
};
int Rhino::getMonth()
{
        return month;
};
//      -----Set/Get gender----- *working*
void Rhino::setGender(char g)
{
        gender = g;
};
char Rhino::getGender()
{
        return gender;
};
//      -----Set/Get tag----- *working*
void Rhino::setTag(int t)
{
        tag = t;
};
int Rhino::getTag()
{
        return tag;
};
//      -----Get children----- *working*
int Rhino::getNumChildren()
{
        return children;
};
//      -----Set/Get mother/father----- *working*
void Rhino::addMother(Rhino * mom)
{
        mother = mom;
        mom->children++;
```

```
};
void Rhino::addFather(Rhino * dad)
{
        father = dad;
        dad->children++;
};

Rhino * Rhino::getMother()
{
        return mother;
};
Rhino * Rhino::getFather()
{
        return father;
};
// -----verfication of gender----- *working*
bool Rhino::isMale()
{
        return this->gender == 'M';
};

void Rhino::print()
{

//      stringstream sMother;
//      stringstream sFather;

//---------------------This solution does not work : Program hangs when sMother &
sFather is declared--------------------------
//      sMother << getMother()->getName();  // cout doesnt work if these two are
uncommented
//      sFather << getFather()->getName();

        //cout << (isMale() ? "Male" : "Female") << " rhino " << getName() << ".
Born " << getMonth() << '/' << getYear() << (getTag() > 0 ? ". Tag " + sTag.str() :
"")
        //      << ((this->getMother() != NULL) ? (". Mother " + sMother.str()) :
(""))
        //      << ((this->getFather() != NULL) ? (". Father " + sFather.str()) :
(""))
        //      << ". Children " << getNumChildren() << ".\n";
//---------------------------------------------------------------------------------
-----------------------------------------

        stringstream sTag;      // sstream for Tag integer
        sTag << getTag();       // Assigning int tag -> sstream sTag

        cout << (isMale() ? "Male" : "Female") << " rhino " << getName() << ". Born
```

```
" << getMonth() << '/' << getYear() << (getTag() > 0 ? ". Tag " + sTag.str() : "");

        switch (getMother() != NULL )           // warning C4144: '!=' : relational
expression as switch expression
        {
        case true:
                cout << ". Mother " << (getMother()->getName());
                break;
        case false:
                break;
        }

        switch (getFather() != NULL)            // warning C4144: '!=' : relational
expression as switch expression
        {
        case true:
                cout << ". Father " << (getFather()->getName());
                break;
        case false:
                break;
        }

        cout << ". Children " << getNumChildren() << ".\n";
}


bool Rhino::isMotherOf(Rhino e)
{
        return (e.getMother() != NULL && e.getMother() == this);
};
bool Rhino::isFatherOf(Rhino e)
{
        return (e.getFather() != NULL && e.getFather() == this);
};
bool Rhino::isParentOf(Rhino e)
{
        return (isMotherOf(e) || isFatherOf(e));
};
bool Rhino::isSisterOf(Rhino e)
{
        return (((e.nickname != this->nickname) && (this->isMale() == false)) &&
(this->father == e.father));    // using isParentOf() does not work
};
bool Rhino::isYounger(Rhino f)
{
        return (((f.getYear() == getYear()) ? (f.getMonth() < getMonth()) :
(f.getYear() < getYear())));
};
```