```
#include <iostream>
using namespace std;

class Teacher{
        string name;
        int age, numOfStudents;
        Teacher * next; // Pointer to next object of teacher
        public:
        Teacher(const string &, int, int); // Constructor
        void print() const;
        const string& getName() const {return name;}
        ~Teacher() { // only to show that the destructor is called
        cout<<" Destructor of teacher" << endl;
        }
        friend class Teacher_list;
};

Teacher::Teacher(const string &new_name,int a,int nos){
        name = new_name;
        age=a;
        numOfStudents=nos;
        next=0;
}
void Teacher::print() const{
        cout <<"Name: "<<name.c_str()<<" Age: "<< age<< endl;
        cout << "Number of Students: " <<numOfStudents << endl;
}

class Teacher_list{ // linked list for teachers
        Teacher *head;
        public:
        Teacher_list(){head=0;}
        bool append(const string &,int,int);
        bool del(const string &);
        void print() const ;
        ~Teacher_list();
};

// Append a new teacher to the end of the list
// if there is no space returns false, otherwise true
bool Teacher_list::append(const string & n, int a, int nos){
        Teacher *previous, *current, *new_teacher;
        new_teacher=new Teacher(n,a,nos);
        if (!new_teacher) return false; // if there is no space return false
        if(head) // if the list is not empty
        {
```

```
        previous=head;
        current=head->next;
        while(current) // searh for the end of the list
        {
        previous=current;
        current=current->next;
        }
        previous->next=new_teacher;
        }
        else // if the list is empty
        head=new_teacher;
        return true;
}
// Delete a teacher with the given name from the list
// if the teacher is not found returns false, otherwise true
bool Teacher_list::del(const string & n){
        Teacher *previous, *current;
        if(head) // if the list is not empty
        {
                if (n.compare(head->getName()) ==0) //1st element is to be deleted
        {
        previous=head; head=head->next; delete previous; return true;
        }
        previous=head;
        current=head->next;
        while( (current) && (n.compare(current->getName()) !=0 )) // searh for the
end of the list
        {
        previous=current; current=current->next;
        }

        if (current==0) return false;
        previous->next=current->next; delete current; return true;
        } //if (head)
        else // if the list is empty
        return false;
}
// Prints all elements of the list on the screen
void Teacher_list::print() const{
        Teacher *tempPtr;
        if (head){
        tempPtr=head;
        while(tempPtr){
        tempPtr->print(); tempPtr=tempPtr->next;
        }
        }
        else cout << "The list is empty" << endl;
        }
        // Destructor
        // deletes all elements of the list
        Teacher_list::~Teacher_list(){
        Teacher *temp;
        while(head) // if the list is not empty
        {
        temp=head; head=head->next; delete temp;
        }
}
// ----- Main Function -----
int main(){
        Teacher_list theList;
        theList.print(); theList.append("Teacher1",30,50);
        theList.append("Teacher2",40,65); theList.append("Teacher3",35,60);
        theList.print();
        if (!theList.del("TeacherX")) cout << " TeacherX not found" << endl;
        theList.print();
        if (!theList.del("Teacher1")) cout << " Teacher1 not found" << endl;
        theList.print();
        return 0;
}
```