LINKED LIST 1

```
class LinkedList {
  ListElement * first;
  public:
    LinkedList();
    void addElement(ListElement * e);
    ListElement * getElement();
    ListElement * removeElement();};
```

```
LinkedList::LinkedList(){ first=NULL; }
void LinkedList::addElement(ListElement * e){
        e->setNextElement(first);
        first=e;
}
ListElement * LinkedList::getElement(){ return first; }
ListElement * LinkedList::removeElement(){
        ListElement * temp=first;
        first=first->getNextElement();
        return temp;
}
```

LINKED LIST 2

```
class ListElement {
  ListElement * next;
  PayLoad * p;
  public:
   ListElement();
   void setNextElement(ListElement * n);
   ListElement * getNextElement();
   void setPayLoad(PayLoad * l);
   PayLoad * getPayLoad();
};
```

```
ListElement::ListElement() {
   next=NULL;
   cout<<"ListElement generated"<<endl;
}
void ListElement::setNextElement(ListElement * n) { next=n; }
ListElement * ListElement::getNextElement() { return next; }
PayLoad * ListElement::getPayLoad() { return p; }
void ListElement::setPayLoad(PayLoad * l) { p=l; }
```

```
class PayLoad{
  public:
    PayLoad();
  friend ostream & operator<<(ostream &sx, const PayLoad &s) {
    sx<<"<Empty PayLoad>";
    return sx;
  }
};
```

```
PayLoad::PayLoad(){
  cout<<"Constructor of PayLoad:"<<*this<<endl;
}
```



```
int main() {
  LinkedList ll;
  for (int i=0; i<3; i++) {
    PayLoad * p=new PayLoad();
    ListElement * le=new ListElement();
    le->setPayLoad(p);
    ll.addElement(le);
  }
  for (int i=0; i<3; i++) {
    ListElement * le=ll.removeElement();
    cout << *(le->getPayLoad()) << endl;
  }
}
```

Output:

```
Constructor of PayLoad:<Empty PayLoad>
ListElement generated
Constructor of PayLoad:<Empty PayLoad>
ListElement generated
Constructor of PayLoad:<Empty PayLoad>
ListElement generated
<Empty PayLoad>
<Empty PayLoad>
<Empty PayLoad>
```



```
class IntPayLoad: public PayLoad {
  int carico;
  public:
    IntPayLoad();
    void load(int i);
    friend ostream & operator<<(ostream &sx, const IntPayLoad &s) {
      sx<<s.carico;
    return sx;
    }
};
```

```
IntPayLoad::IntPayLoad(){
      cout<<"Constructor of IntPayLoad"<<endl;
    }
    void IntPayLoad::load(int i){
      carico=i;
    }
```

```
int main(){
  LinkedList ll;
  for (int i=0; i<3; i++) {
    IntPayLoad * p=new IntPayLoad();
    p->load(i);
    ListElement * le=new ListElement();
    le->setPayLoad(p);
    ll.addElement(le);
  }
  for (int i=0; i<3; i++) {
    ListElement * le=ll.removeElement();
    cout << *((IntPayLoad *)(le->getPayLoad())) << endl;
  }
}
```



```
class StringPayLoad: public PayLoad {
  String carico;
  public:
    StringPayLoad();
    void load(String * s);
   friend ostream & operator<<(ostream &sx,
                               const StringPayLoad &s) {
      sx<<s.carico;
    return sx;
    }
};
```

```
StringPayLoad::StringPayLoad(){
      cout<<"Constructor of StringPayLoad"<<endl;
   }
   void StringPayLoad::load(String * s){
     carico=*s;
   }
```

```
int main() {
  LinkedList ll;
  char s[]={'S','x',0};
  for (int i=0; i<3; i++) {
    s[1]=(char)(65+i);
    StringPayLoad * p=new StringPayLoad();
    p->load(new String(s));
    ListElement * le=new ListElement();
    le->setPayLoad(p);
    ll.addElement(le);
  }
  for (int i=0; i<3; i++) {
    ListElement * le=ll.removeElement();
    cout << *((StringPayLoad *)(le->getPayLoad())) << endl;
  }
}
```



```
class DoubleStringPayLoad: public PayLoad {
  String carico1;
  String carico2;
  public:
    DoubleStringPayLoad();
    void load(String * s, String * t);
    friend ostream & operator<<(ostream &sx,
                const DoubleStringPayLoad &s) {
      sx<<s.carico1<<" - "<<s.carico2;
    return sx;
    }
};
```

```
DoubleStringPayLoad::DoubleStringPayLoad(){
    cout<<"Constructor of StringPayLoad"<<endl;
  }
  void DoubleStringPayLoad::load(String * s, String *t){
    carico1=*s;
    carico2=*t;
  }
```

```
int main() {
  LinkedList ll;
  char s[]={'S','x',0};
  char t[]={'T','x',0};
    for (int i=0; i<3; i++) {
    s[1]=(char)(65+i);
    t[1]=(char)(65+i);
    DoubleStringPayLoad * p=new DoubleStringPayLoad();
    p->load(new String(s),new String(t));
    ListElement * le=new ListElement();
    le->setPayLoad(p);
    ll.addElement(le);
  }
  for (int i=0; i<3; i++) {
    ListElement * le=ll.removeElement();
    cout << *((DoubleStringPayLoad *)(le->getPayLoad())) << endl;
  }
}
```


Output:

```
Constructor of PayLoad:<Empty PayLoad>
Constructor of StringPayLoad
ListElement generated
Constructor of PayLoad:<Empty PayLoad>
Constructor of StringPayLoad
ListElement generated
Constructor of PayLoad:<Empty PayLoad>
Constructor of StringPayLoad
ListElement generated
SC - TC
SB - TB
SA - TA
```