In [1]:

```
from rdkit import Chem
from rdkit.Chem import AllChem, rdMolDescriptors, rdChemReactions, Draw
from rdkit.Chem.Draw import IPythonConsole
from rdkit import RDLogger
from rdkit import rdBase
rdBase.DisableLog('rdApp.error')

#dopts = Draw.rdMolDraw2D.MolDrawOptions()
#dopts.prepareMolsForDrawing = True
```

Bischler-Napieralski reaction

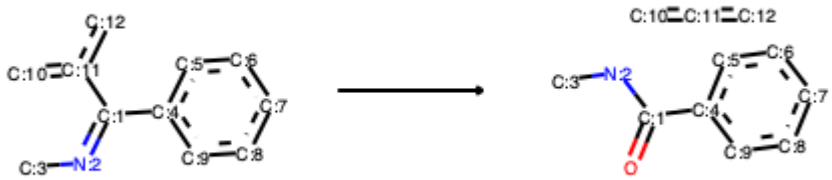
Error: 2-component incorrect disconnection

Template ID: f90c010d7788dcda116c0ca6c4900611b1947e783dc8f787933b192b

In [2]:

```
# Reaction template
rxn = AllChem.ReactionFromSmarts('([C:3]-[N;H0;D2;+0:2]=[C;H0;D3;+0:1](-[c:4]1:[c:5]
rxn
```

Out[2]:

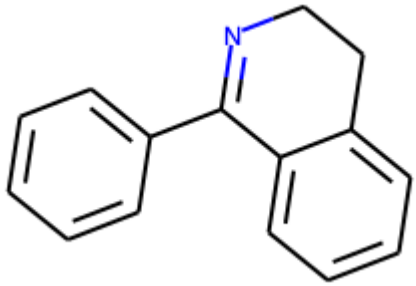


In [3]:

```
# Run retro template
prod = Chem.MolFromSmiles("c1c2c(ccc1)CCN=C2c3ccccc3")
a = rxn.RunReactants((prod,))
print(len(a))
prod
```

```
8
```

Out[3]:

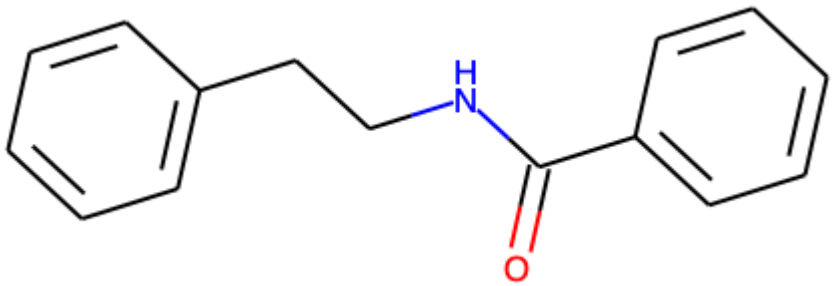


In [4]:

```
# Correct disconnection
A = a[0]
print("Product has", len(Chem.GetMolFrags(A[0])), "fragment(s)")
A[0]
```

```
Product has 1 fragment(s)
```

Out[4]:

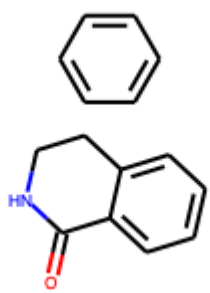


In [5]:

```
# Wrong disconnection
B = a[4]
print("Product has", len(Chem.GetMolFrags(B[0])), "fragment(s)")
B[0]
```

```
Product has 2 fragment(s)
```

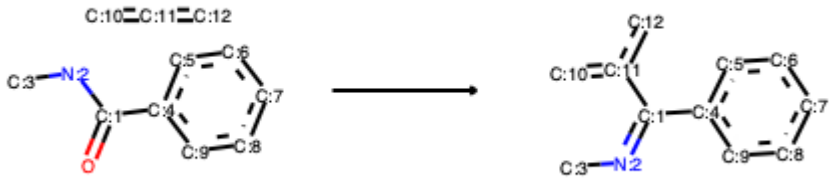
Out[5]:



In [6]:

```
# forward reaction
Frxn = AllChem.ReactionFromSmarts('(O=[C;H0;D3;+0:1](-[NH;D2;+0:2]-[C:3])-[c:4]1:[c:
Frxn
```

Out[6]:

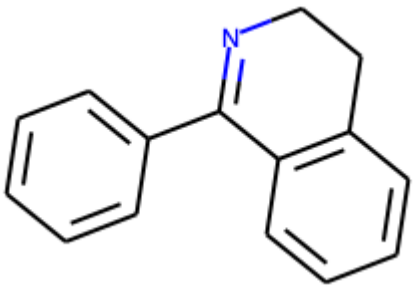


In [7]:

```
# Check forward reaction
p1 = Frxn.RunReactants(A)
print(len(p1))
p1[0][0]
```

```
20
```

Out[7]:

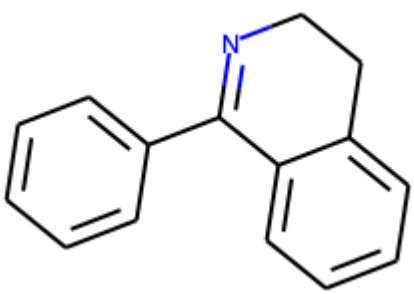


In [8]:

```
# Check forward reaction
p2 = Frxn.RunReactants(B)
print(len(p2))
#img = Draw.MolsToGridImage([m[0] for m in p1], molsPerRow=4, drawOptions=dopts)
p2[0][0]
```

```
24
```

Out[8]:



Note: Although both disconnections can formally lead to the desired product in a forward manner, "A" disconnection defines an intramolecular cyclisation reaction (CORRECT) while "B" disconnection is basically a two-component reaction (INCORRECT)

Bromo N-alkylation

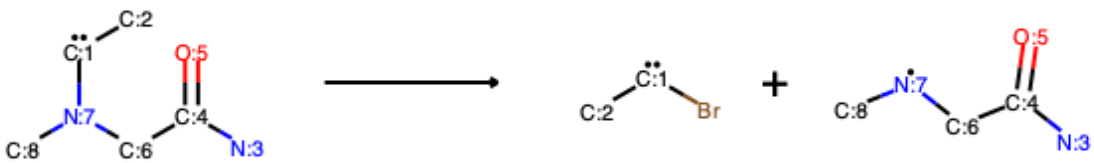
Error: Product cannot be regenerated

Template ID: 10862e3979fa2dbdadfd071c924544659dd19abdfd1363ef90c38776

In [9]:

```
# Reaction template
rxn = AllChem.ReactionFromSmarts('([#7:3]-[C:4](=[O;D1;H0:5])-[C:6]-[N;H0;D3;+0:7](-
rxn
```

Out[9]:

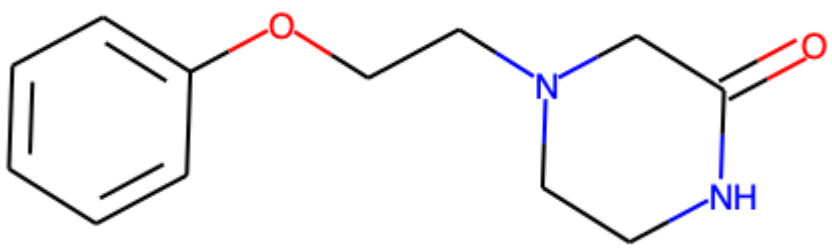


In [10]:

```
# Run retro template
prod = Chem.MolFromSmiles("c1ccc(cc1)OCCN2CCNC(C2)=O")
a = rxn.RunReactants((prod,))
print(len(a))
prod
```

2

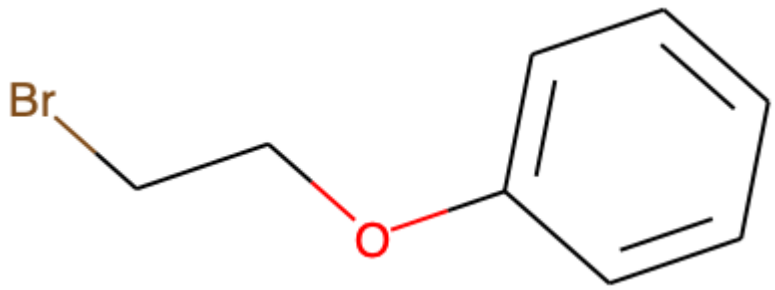
Out[10]:



In [11]:

```
# Correct disconnection
A = a[1]
A[0]
```

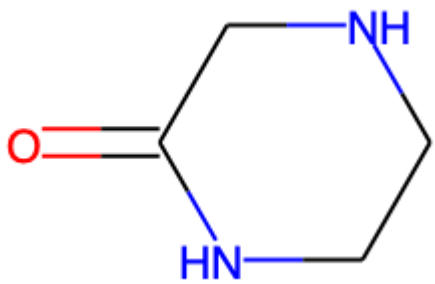
Out[11]:



In [12]:

```
A[1]
```

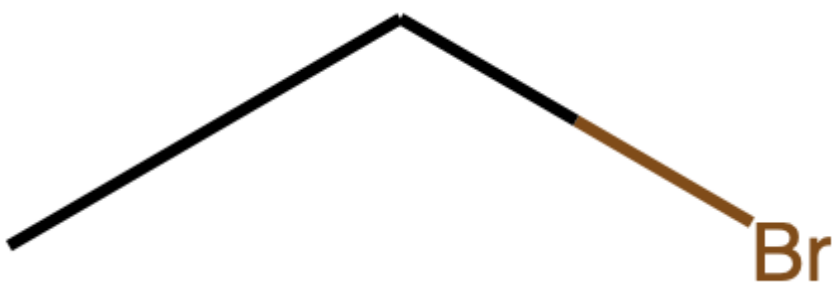
Out[12]:



In [13]:

```
# Wrong disconnection
B = a[0]
B[0]
```

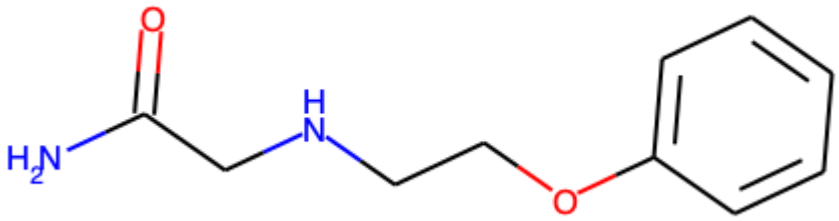
Out[13]:



In [14]:

```
B[1]
```

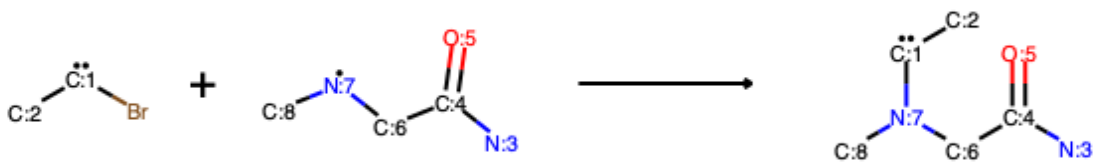
Out[14]:



In [15]:

```
# forward reaction
Frxn = AllChem.ReactionFromSmarts('(Br-[CH2;D2;+0:1]-[C:2]).([#7:3]-[C:4](=[O;D1;H0:
Frxn
```

Out[15]:

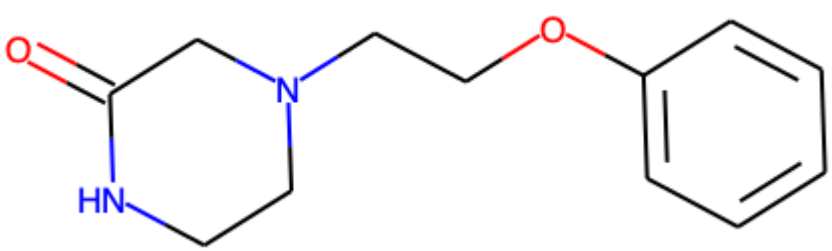


In [16]:

```
# Check forward reaction
p1 = Frxn.RunReactants(A)
print(len(p1))
p1[0][0]
```

```
1
```

Out[16]:

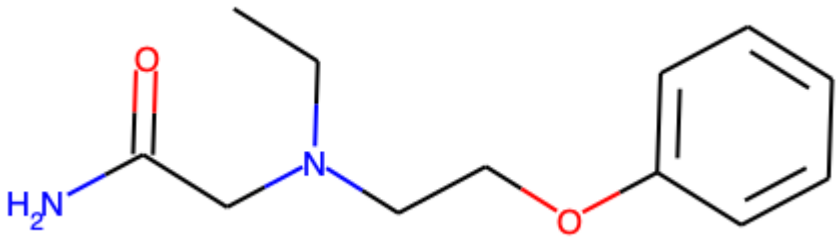


In [17]:

```
# Check forward reaction
p2 = Frxn.RunReactants(B)
print(len(p2))
p2[0][0]
```

```
1
```

Out[17]:



Note: Only the A disconnection can lead to the desired product in a forward manner (CORRECT) while B disconnection is basically incorrect