CPS 100 - Program Design and Analysis II
Dr. Susan Rodger
Section: Templates (handout)

Read Chap. 2 & 3 in Weiss.

More on Pointers

Use a debugger: gdb or ddd

ddd - graphical layout of linked list

Common Mistakes

```
struct Node
{
   int info;
   Node * next;
};
```

1. Obtaining Dynamic Memory

```
Node * cur;
if (cur != NULL)
   cur -> info = 7;
```

2. Dangling (Leaking) Memory

```
Node * temp = new Node;
Node * temp2 = new Node;
temp = temp2;
```

3. Dereference NULL pointer

```
while (temp->next != NULL)
{
   cout << temp->info << " ";
   temp = temp->next;
}
```

4. Deleting Dynamic Memory

```
Node * cur = new Node(3);  // assume constructor exists
Node * temp1 = cur;
delete cur;
delete temp1;
```

Template Functions

Template allows reuse of code.

Pattern for function for multiple types.

```
template <class Type>
void Swap(Type & one, Type & two)
{
   Type temp = one;
   one = two;
   two = temp;
}
```

Compile?

Use?

```
string a = "Knuth";
string b = "Turing";
int x = 78;
int z = 46;

Swap(a,b);
Swap(x,z);
```

Template Class

Avoids multiple class definitions

Format: (T - variable, Cn - class name

```
template <class T>
class cn
{
...
}
```

Use:

```
  Cn<int> A;
  Cn<double> B;
```

Example: Vector class

This is the vector class from CPS 6 - see vector.cc

Use:

```
Vector <int> Ids;
Vector <string> Names;
```

Observations in Vector class

- template
  How do you write a templated function that appears outside the class definition?
- assert
- this
- const
  - const reference
  - const function
  - const return value
- operator