1. Complete the code for the following.

```
template <class T>
bool inList(const SNode<T>* head, T val)

{
      SNode<T> *cursor = head;
      while (cursor != NULL)
      {
            if (cursor->elem == val)
                  return 1;
            else
                  cursor = cursor->next;
      }

      return 0;
}
```

2. Complete the code for the following:

```
template <class T>
bool inList(const SNode<T>* header,
            const SNode<T>* trailer, T val)

{
      SNode<T> *cursor = header->next;
      while (cursor != trailer)
      {
            if (cursor->elem == val)
                  return 1;
            else
                  cursor = cursor->next;
      }

      return 0;
}
```

3. Rewrite the code for the doubly-linked list function insertAfter without using the constructor.

```
void insertAfter(Dnode<T>* header,
                 DNode <T>*trailer,
                 Dnode<T>* current, T newval)
{
      assert (current != trailer);
      Dnode<T> *newnode = new Dnode<T>();
      newnode->elem = newval;
      newnode->prev = current;
      newnode->next = current->next;
      current->next->prev = newnode;
      current->next = newnode;
}
```

4. Complete the code for the following:

```
template <class T>
void File2List(istream &in, T stopval
               Dnode* &h, Dnode* &t)

{
   //using DLL given in LinkedLists.pptx
   h = new Dnode<T>();
   t = new Dnode<T>();
   Dnode<T> *header = h;
   Dnode<T> *trailer = t;
   Dnode<T> *current= header;
   T inc;
   while (in>>inc)
   {
     if (inc != stopval)
     {
          current->next = new DNode<T> (inc, current,
current->next);
          current=current->next;
          current->next->prev = current;
     }
     else
     {
          current->next = trailer;
          current->next->prev=current;
          break;
     }
   }
}
```

5. Complete the code for the following:

```
template <class T>
int CircleList::size()

{
      //using CircleList class given in LinkedLists.pptx

      if (empty())
            return 0;
      Cnode<T> *getsize = cursor->next;
      int size = 1;   //start at 1 in case only 1 node
      while (getsize != cursor)
      {
            getsize = getsize->next;
            size++;
      }

      return size;
}
```