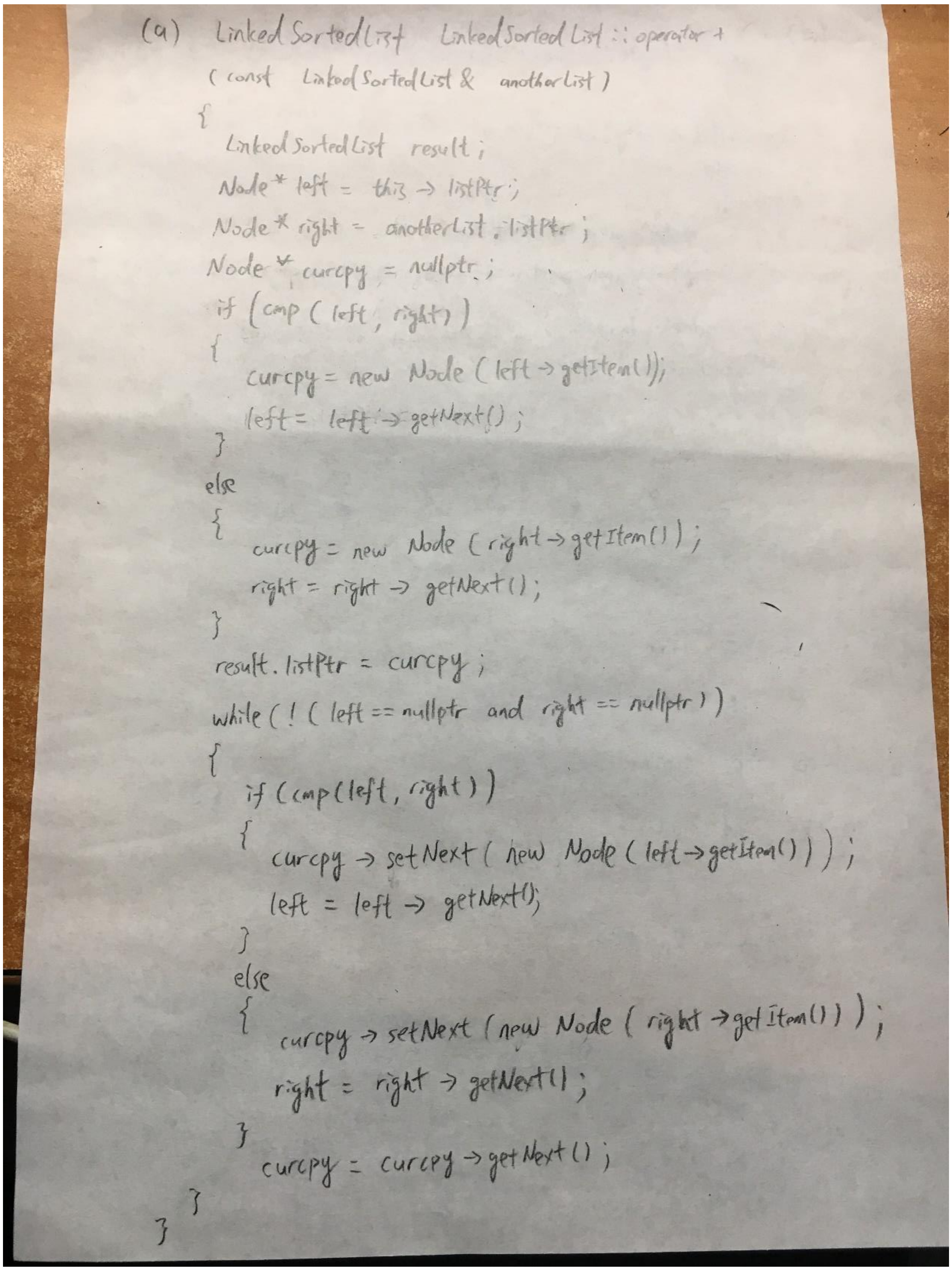
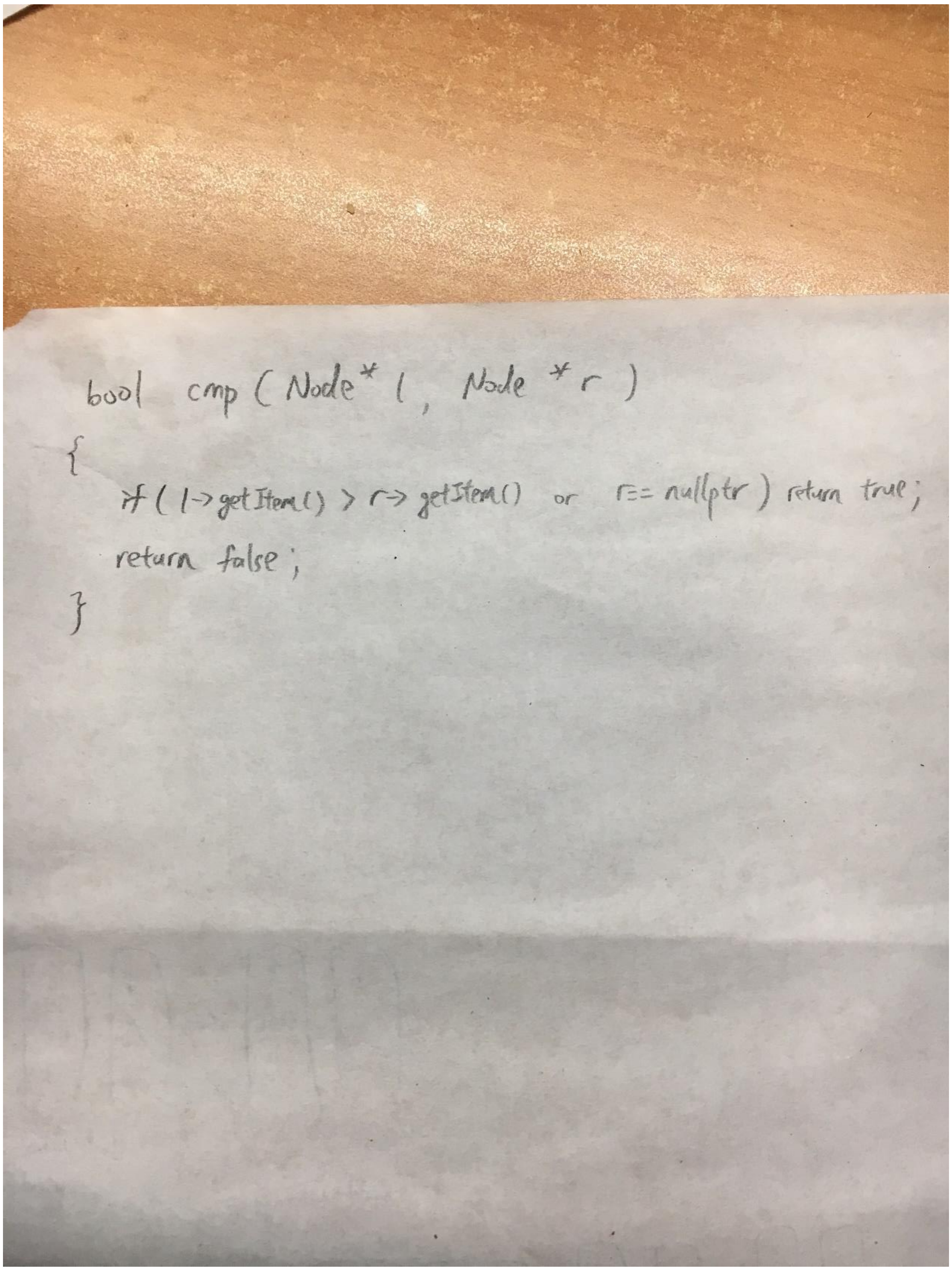
DSAP Homework 6 手寫

B05703100 財金三 郭仲嘉

第一題



```
(a) LinkedSortedList LinkedSortedList :: operator +
    ( const LinkedSortedList & anotherList )
    {
      LinkedSortedList result;
      Node* left = this -> listPtr;
      Node* right = anotherList . listPtr;
      Node * curcpy = nullptr;
      if (cmp ( left, right) )
      {
        curcpy = new Node ( left -> getItem());
        left = left -> getNext();
      }
      else
      {
        curcpy = new Node ( right -> getItem() );
        right = right -> getNext();
      }
      result . listPtr = curcpy;
      while ( ! ( left == nullptr and right == nullptr ) )
      {
        if (cmp(left, right))
        {
          curcpy -> setNext ( new Node ( left -> getItem() ) ) ;
          left = left -> getNext();
        }
        else
        {
          curcpy -> setNext (new Node ( right -> getItem() ) ) ;
          right = right -> getNext();
        }
        curcpy = curcpy -> getNext();
      }
    }
```



```
bool cmp ( Node* l , Node *r )
{
    if ( l->getItem() > r->getItem()  or  r== nullptr ) return true;
    return false;
}
```

(b)
```
void display (Queue aQueue)
{
  if(aQueue.isEmpty()) return;
  while(1)
  {
    cout << aQueue.peekFront();
    aQueue.dequeue();
    if (aQueue.isEmpty()) return;
    cout << ",";
  }
}
```

(c)
```
void deque:: addfront( int newEntry )
{
  Node* newNodePtr = new Node(newEntry)
  if (isEmpty()) backPtr = newNodePtr;
  else
  {
    newNodePtr-> setNext( frontPtr );
  }
  frontPtr = newNodePtr;
}
```

(d)

```
void deque :: remove_back()
{
    if (isEmpty()) return;
    else
    {
        Node* prev = frontPtr;
        if (! prev->getNext() == backPtr ) prev = prev.getNext();
        delete backPtr;
        backPtr = prev;
    }
}
```