Programming Exercises

1. The linked list example in the notes adds new nodes at the head of the list. Rewrite the program so that nodes are added at the tail. The output of your program should be:

```
(1,3)
(2,4)
(3,5)
(4,6)
(5,7)
```

The original program was:

```
#include<iostream>
using namespace std;

class item {
public:
  int val1,val2;
  item * next;
};

int main() {
   item * head = NULL;
   int i;
   for(i=1;i<=5;i++) {
      item * curr = new item;
      curr->val1 = i;
      curr->val2=i+2;
      curr->next  = head;
      head = curr;
   }
   while(head) {
     cout << "("<<head->val1 << ","<<head->val2<<")"<<endl;
      head = head->next ;
   }
}

/****Output**********
(5,7)
(4,6)
(3,5)
(2,4)
(1,3)
********************/
```

2. Notice that in the example above, there is no linked list class at all. There are only nodes, defined in the class item. The linked list is the pointer to the first node.
Re-write the code above to encapsulate the linked list as its own class with helper functions. For instance,

```
class myList{
  public:
```

```
item* head; //Data
void display(); //Go through the list and display the data in each node of the list
void add_at_head(item); //add item at the head of the list
void add_at_tail(item); //add item at the tail of the list
void add_at_pos(int pos, item i);
//add item i at position pos, where position starts at 0 which is the  head.
int size(); //returns the number of elements in the list
}
```

To test, you can use something like:

```
int main(){
item i[20];
for(int j=0;j<20;j++){          //or you could write a constructor for item class
i[j].val1=j;
i[j].val2=j+2;
}
myList ML;
ML.add_at_end(i[0]);
ML.add_at_tail(i[1]);
ML.add_at_head(i[4]);
ML.add_at_head(i[7]);
ML.display();
}
```