CSY2006 Week 19

Linked List Exercises

1. What output is produced by the following C++ code segment on execution?

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

struct nodeType
{
    int info;
    nodeType *link;
};

int main()
{
    nodeType *list, *ptr;
     list = new nodeType;
    list -> info = 10;
    ptr = new nodeType;
    ptr -> info = 13;
    ptr -> link = NULL;
    list -> link = ptr;
    ptr = new nodeType;
    ptr -> info = 18;
    ptr -> link = list -> link;
    list -> link = ptr;
    cout << list -> info << " " << ptr -> info << " ";
    ptr = ptr -> link;
    cout << ptr -> info << endl;
    return 0;
}
```

Answer:

2. What output is produced by the following C++ code segment?

```
struct nodeType
{
   int info;
   nodeType *link;
};

int main()
{
   nodeType *list, *ptr;
   list = new nodeType;
   list->info = 20;
   ptr = new nodeType;
   ptr->info = 28;
   ptr->link = NULL;
   list->link = ptr;
   ptr = new nodeType;
   ptr->info = 30;
   ptr->link = list;
   list = ptr;
   ptr = new nodeType;
   ptr->info = 42;
   ptr->link = list->link;
   list->link = ptr;
   ptr = list;
   while (ptr != NULL)
   {
          cout << ptr->info << endl;
          ptr = ptr->link;
   }
   system("PAUSE");
   return 0;
}
```

Answer:

3. What output is produced by the following C++ code segment on execution?

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

struct nodeType
{
     int info;
     nodeType *link;
};

int main()
{
     nodeType *list, *ptr;
      list = new nodeType;
     list -> info = 21;
     ptr = new nodeType;
     ptr -> info = 45;
     ptr -> link = NULL;
     list -> link = ptr;
     ptr = new nodeType;
     ptr -> info = 79;
     ptr -> link = list -> link;
     list -> link = ptr;
     cout << list -> info << " " << ptr -> info << " ";
     ptr = ptr -> link;
     cout << ptr -> info << endl;
     return 0;
}
```

Answer:

4. What output is produced by the following C++ code segment?

```
struct nodeType
{
   int info;
   nodeType *link;
};

int main()
{
   nodeType *list, *ptr;
   list = new nodeType;
   list->info = 34;
   ptr = new nodeType;
   ptr->info = 56;
   ptr->link = NULL;
   list->link = ptr;
   ptr = new nodeType;
   ptr->info = 73;
   ptr->link = list;
   list = ptr;
   ptr = new nodeType;
   ptr->info = 99;
   ptr->link = list->link;
   list->link = ptr;
   ptr = list;
   while (ptr != NULL)
   {
          cout << ptr->info << endl;
          ptr = ptr->link;
   }
   system("PAUSE");
   return 0;
}
```

Answer:

Lab Exercises:

1. Design your own linked list class to hold a series of integers. The class should have member functions for appending, inserting, and deleting nodes. Don't forget to add a destructor that destroys the list. Demonstrate the class with a driver program. Add a print member function. The function should display all the values in the linked list. Test the class by starting with an empty list, adding some elements, and then printing the result out.

2. Modify the linked list class above to add a copy constructor. Test your class by making a list, making a copy of the list, and then displaying the values in the copy.

3. Modify the linked list above to insert a new item at a specified position. A position of 0 means that the value will become the first item on the list, a position of 1 means that the value will become the second item on the list, and so on. A position equal to or greater than the length of the list means that the value is placed at the end of the list.

4. Modify the linked list above to add a member function named reverse that rearranges the nodes in the list so that their order is reversed.

5. Create a linked list template based on the previous exercises and include a member function named search that returns the position of a specific value in the linked list. The first node in the list is at position 0, the second node is at position 1, and so on. If x is not found on the list, the search should return -1. Test the new member function using an appropriate driver program.