# CS 341
# H. W. #2
# Sorted Linked Lists

## Due March 27

For this machine problem you will read in a sequence of non-zero integers, terminated by 0 and store them in a (singly) linked list. In particular you are to have two lists one for even integers and one for odd integers. You are to insert the links into the list in such a way that the list is sorted from smallest to largest at all times. You are to then print out the list. Next you are two merge the two lists into a single sorted list which should be sorted from smallest to largest. This is to be done by removing links from the even and odd list and inserting them into the merge sorted list. Merging will be discussed in class and you are to follow the method given in class. You should print out the merged list and finally recycle all of the links.

You MUST write the program by supplying the missing code to the partial program supplied.

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

struct link {
     int value;
     link * next;

     link() { value = 0; next = 0; }
     link(int _value) {value = _value; next = 0; }
};

void printList( link * );

int main(){
     link * eventop = 0;
     link * oddtop = 0;
     link * mergetop = 0;
```

```
     link * mergebottom;
     link * previous = 0;
     link * current = 0;
     link * lptr = 0;

     int invalue;

     cout << "Input an integer, enter 0 to terminate: " << flush;
     cin >> invalue;

     while ( /* test condition goes here */ ) {
/* This while loop builds the two sorted linked lists. */


     cout << "Input an integer, enter 0 to terminate: " << flush;
     cin >> invalue;

     }

printList( oddtop );
cout << "\n\n";
printList( eventop );
cout << "\n\n";

     mergetop = mergebottom = 0;
     while ( /* test condition goes here */ ) {
/* Code to build the merged list goes here.
   mergetop should point to the top of the list
   mergebottom whould point to the bottom of
   the list */

     }
     printList ( mergetop );

     while ( mergetop ) {
          lptr = mergetop;
          mergetop = mergetop->next;
          delete lptr;
```

```
     }
return 0;
}


void printList( link * list){

/* Your code goes here. */

}
```

Next you are to rewrite your program using two arrays in place of pointers and dynamically allocated memory as was discussed in class at the start of the section on linked lists. In all other ways the second program should be the same as the first. Again you are to fill in the missing code in the partial program provided. Be sure that all memory accessed from your arrays is allocated by `nextLocation()` which should take the next available location from the top of the list.

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

const int ArraySize = 32;

int value[ArraySize] = {0};
int next[ArraySize] = {0};
int top = 1;

void initializeArrays();

int nextLocation();

void printlist( int );

void recycle( int );
```

```
void dump();

int main() {
     initializeArrays();

     int eventop = 0;
     int oddtop = 0;
     int mergetop = 0;
     int mergebottom = 0;
     int current = 0;
     int previous = 0;
     int lptr = 0;

     int invalue;

     cout << "Input an integer, enter 0 to terminate: " << flush;
     cin >> invalue;

     while ( /* test condition goes here */ ) {
/* This while loop builds the two sorted linked lists. */


     cout << "Input an integer, enter 0 to terminate: " << flush;
     cin >> invalue;

     }
printlist( oddtop );
cout << "\n---------\n";
printlist( eventop );
cout << "\n---------\n";
cout << "oddtop = " << oddtop << endl;
cout << "eventop = " << eventop <<endl;
dump();

     mergetop = mergebottom = 0;
     while ( /* test condition goes here */ ) {
/* Code to build the merged list goes here.
   mergetop should point to the top of the list
```

```
   mergebottom whould point to the bottom of
   the list */

     }
cout << "\n---------\n";
printlist( mergetop );

while ( mergetop ) {
     lptr = mergetop;
     mergetop = next[mergetop];
     recycle( lptr );
     }
cout << "\n---------\n";
dump();
}

void initializeArrays() {
    for ( int i = 1; i < ArraySize; i++) {
         next[i] = i + 1;
    }
    next[ArraySize - 1] = 0;
}

int nextLocation() {
/* Your code goes here. */
}


void printlist( int list ) {
/* Your code goes here. */
}

void recycle( int  link) {
/* Your code goes here. */
}

void dump() {
     cout << "top = " << top << endl;
```

```
     for ( int i = 0; i < ArraySize; i++ ) {
          cout << "[ " << i << " ]" << "value = "
<< value[i] << " next = " << next[i] << endl;
     }
}
```

Use the following test data:

```
1, 2, 3, 4, 5, 6, 6, 7, 7, 5, 4, 3, 2, 1, 0

1, 2, 3, 4, 5, 6, 7, 8, 9, 0

9, 8, 7, 6, 5, 4, 3, 2, 1, 0

2, 2, 2, 2, 2, 2, 2, 0

1, 3, 5,  7, 9, 7, 5, 3, 1, 0
```