CS 2005 Techniques of Programming
Craig E. Wills
Given: Friday, Dec 2, 1994

WPI, B Term 1994
Exam 2 (100 pts)

NAME:______________________________

LAB TIME:______________________________

This is a closed book (and notes) examination. Answer all questions on the exam itself. Take the number of points assigned to each problem and the amount of space provided for your answer as a measure of the length and difficulty of the expected solution. The exam totals 100 points.

SCORE:________________________________

1. (15 points) Indicate whether the following statements are true or false (write True or False, not T or F).

__________ In a doubly linked list, two pointers are maintained—one pointer to the next node in the list and another pointer to the previous node in the linked list.

__________ If a set of data items is stored using a linked list then only a linear search and not hashing or a binary search can be used to search for items.

__________ The worst case search time for a binary search is O(log n) where n is the number of elements to search.

__________ The worst case search time for hashing is O(n) where n is the number of elements to search.

__________ Given that space is not a problem, hashing will yield better average case search time than either a binary or linear search.

2. (12 points) In project two on maze traversals your group implemented four maze traversal strategies (fix, dir, goal and gdir) all using a stack (ignore the “best” strategy for this question).

   (a) Select one of the traversal strategies (clearly indicate which one) and show the output for the following maze (including dead ends if need be). You do not need to print the length of path, number of dead ends, etc.

```
#######
#     #
#  #> #
#  #  #
## ## #
#<    #
#######
```

   (b) What was the basic item type pushed on and popped from the stack?

   (c) *Briefly explain* what was done within a traversal strategy when a point had no free neighbors.

3. (10 points) You are using a hash table to store a set of integers using the following function to compute the hash value for an integer.

```
int Hash(int value)
{
    return(value%10);
}
```

Show where each of the following integers are inserted in the hash table of 10 entries if this hash function is used with linear probing (closed hashing) to resolve collisions. Be sure and indicate the hash value for each value.

```
1110
1129
3145
2309
2175
```

0	1	2	3	4	5	6	7	8	9

4. (9 points) Assume each of the following expressions is the performance of various algorithms for an operation (such as comparisons in sorting) where n is the size of the problem (such as the number of elements). Give the "big oh" notation for each of these expressions.

   (a) $2n^3 + 5n + 1$

   (b) $n^2/2 + 5n$

   (c) $\log n + n$

5. (13 points) Assume we have a linked list implementation for a queue of integers using the following declarations for the queue and a node within the queue.

```
struct nodetag {        /* node definition */
    int item;
    struct nodetag *pNodeNext;
};

struct queue {            /* queue definition */
    struct nodetag *pNodeFront;      /* front of the queue list */
    struct nodetag *pNodeRear;       /* rear of the queue list */
};
```

Write the code for the function *AddQueue*, which is passed a pointer to the queue structure (already initialized and possibly containing values) and an integer to add to the list. The function *MakeNode()* allocates space for the node, filling in the *item* field with the value passed and setting the *pNodeNext* field to NULL.

```
int AddQueue(int value, struct queue *pqueue)
{
    struct nodetag *pNode;

    if ((pNode = MakeNode(item)) == NULL)
        return(-1);        /* error */
    else {
        /* your code goes here */

    }
    return(0);            /* successfully added */
}
```

6. (13 points) Using the same data structures as the previous question, write the function *SumValue()*, which is passed a pointer to a queue structure. The function should return the sum of all integers in the queue.

```
int SumValue(struct queue *pqueue)
{
    /* your code goes here */







}
```

7. (13 points) In each of the following scenarios (a-c) indicate whether a *linked list* or an *array* should be used for storing a list of values. Briefly explain why.

   (a) Random access to any value in the list needs to be fast.

   (b) The number of values contained in the list is unknown before the list is created.

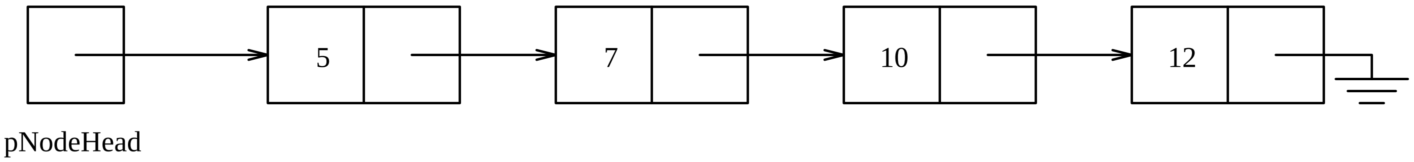
   (c) The values need to be sorted with many insertions and deletions to the list.

   (d) Both an array and a linked list implementation can have space overhead (space not used for actually storing values) for storing a list of elements. Briefly explain what kind of space overhead each approach has.

8. (15 points) Assume we have a linked list of integers using the following declaration for a node with the head of the linked list stored in the variable `pNodeHead`.

```
struct nodetag {
    int item;
    struct nodetag *pNodeNext;
};

struct nodetag *pNodeHead;     /* head of list */
```



(a) A sorted linked list is built as shown. Assume that the number 10 is to be deleted from the linked list. Indicate on the diagram above the values of the pointers after this node has been deleted from the list.

(b) What should happen to the node associated with the number 10?

(c) Now write the code segment to delete a node with the integer *value* from a linked list pointed to by `pNodeHead`. You can ignore the cases where the list is empty or the integer to delete is at the head of the list. Show declarations for any additional variables you use. Just write a code segment, not an entire procedure or program.