CS 300 Exam 3 Review Dynamic memory, Linked Lists, Generic Programming, Hash Tables

1. Given the following program, show exactly what is printed on the screen after execution.

```
void reviewOne(int **hTheInt, int *pTheInt, int theInt)
{
  printf("1: %d %d %d\n", **hTheInt, *pTheInt, theInt);

  **hTheInt =  4;
  *pTheInt = 2;
  theInt = 6;     /* MARK */
}

int main()
{
 int intOne = 1;
 int intTwo = 2;
 int *pTheInt = &intTwo;

 reviewOne(&pTheInt,&intOne, intTwo);
 printf("2: %d %d %d\n", *pTheInt, intOne, intTwo);
}
```

2. Draw a picture of memory to show exactly what data is stored where after the line marked above is executed.

3. Write the following function that will print the following linked list from start to finish.

```
typedef struct Node {
{
  int data;
  struct Node *psNext;
} Node;
```

void printForward(Node *psHead);

4. Write a function that will print the list backwards.

void printBackwards(Node *psHead);

5. Write a function that will remove (and free) every third node from the list:
1->12->6->9->0->6->7->8 will become 1->12->9->0->7->8

void removeThird(Node *psHead);

6. Write a function that will reverse a list:
1->12->6->9->0->6->7->8 will become 8-> 7-> 6-> 0-> 9-> 6-> 12-> 1
void reverse(Node **hHead);

7. Write a function that will merge the two given lists into one list. Take the first node from the left list followed by the first node in the right list and so on. If either list is bigger than the other list, just put the extra nodes at the end without interleaving.

   void printMerge(Node *psLeft, Node *psRight, Node **hNewList);

   ex: psLeft: 1->2->9 psRight: 3->7->19->100
   result: hNewList: 1->3->2->7->9->19->100

8. Write a function, visit, that will call a function pointer on each node in a list. The function pointer, when called, should be passed a node in the list.

   void visit(Node* psHead, (void)(*function)(Node* psNode));

9. Use the function from 8. to print every element of the list.

10. Use the function from 8. to print every element of the list that contains an even integer.

11. Call qsort to sort an array of floats.

12. Finish the Labs from class.
    1. Dynamic Memory Lab
    2. Valgrind Lab
    3. Function Pointer Lab

13. Explain what a collision is during hashing.

14. Describe how linear probing can lead to poor access times for a Hash Table.

15. Hash the keys M6, G7, Q21, Y77, R19, Z18, and F2 using the hash formula h(Kn) = n mod 7 with the following collision handling technique:

    1. linear probing
    2. quadratic probing
    3. chaining
       1. unsorted chains
       2. sorted chains

    4. Compute the average number of probes to find an arbitrary key for all methods

16. The tool gpref (http://www.gnu.org/software/gperf/) will generate C code that produces a perfect hash function. However, gpref requires that you provided every key you will ever hash. Why is this necessary?