1. (24 points) What do each of the following expressions evaluate to, for all variables defined? For pointer variables, say which variable’s address they contain. If any of the expressions are invalid, say so and why.

   a) char x, y, z, *p, *q;
      x = 'x';
      y = 'y';
      p = &x;
      q = &y;
      z = *q – *p;

   b) int x;
      int *p, *q;
      *p = -2;
      *q = 4;
      x = *p + *q;

   c) char *s = “hiccup”, ch1, ch2, *p, *q;
      p = s;
      q = s+5;
      ch1 = * (++p);
      ch2 = * (q--);
      *q = *(s + 6);

      Also fill in the memory diagram for s:
2. (12 points) What does the following segment of C++ code print? You must show your work in order to receive partial credit if your solution is incorrect.

```cpp
int i, N;
char s[] = "odyssey", *f, *b, temp;

N = strlen(s);
for(f = s, b=s+N-1; f < b; f++, b--)
{
    temp = *f;
    *f = *b;
    *b = temp;
}
cout << s;
```
4. (32 points) a) Write a function that counts the number of distinct digits from an integer. Hint: Use char* `itoa(int N)` to convert int `N` to a C-string (null-terminated character array) representation. Loop through the resulting string, and when you encounter a given character for the first time add 1 to the count.

b) Write the output produced from main below.

```c
int countDistinctDigits(int N)
{

int main()
{
    cout << "1112 has " << countDistinctDigits(1112) << " distinct digits.\n";
    cout << "1252 has " << countDistinctDigits(1252) << " distinct digits.\n";
    return 0;
}
```
4. (24 points) Write a program that does the following. When executed, it requires one command line argument that is the name of a text file. The text should contain a single integer on each line and have an unknown number of lines. Your program should read the text file and print the numbers in reverse order. *Hint: Some functions and variables that you will find useful are: argc, argv[], new, delete.* Note that you will be heavily deducted points if you use a fixed length array to solve this problem. See example input/output below.

$ ./a nums.txt
11
7
5
3
2
5. (24 points) a) Write a function, \texttt{int minArray(int x[], int N)} that finds the minimum value in an array of integers, x, with N elements.

b) Suppose a pond starts on day 1 with 3 lilies. Each day, the number of lilies doubles and then one dies (due to a frog being too rough with it!) Write a function, \texttt{int lilies(int N)}, that calculates the number of lilies on day N.

c) Write your own version of a string comparison function: \texttt{int stringCompare(string s, string t)}. You cannot use the string comparison operators, e.g. <, >, ==, but must create your own comparison by iterating through the characters of the strings. Your function should return -1 if s < t, 1 if s > t, and 0 if s == t.

Example cases:
\texttt{stringCompare("hey", "ho") = -1}
\texttt{stringCompare("ho", "hey") = 1}
\texttt{stringCompare("ho", "ho") = 0}
6. Consider a LinkedList struct as developed in L18.
   a) Write a function `void consolidate(LinkedList &A, LinkedList &B)` that appends all Node*'s from B to the end of A. The result will be that no new memory is allocated, but the existing Node*'s of B become part of A, and B become an empty LinkedList.

   b) Write a function `void copy(LinkedList &A, LinkedList &B)` that creates a new copy of all the Nodes from A and puts them into B. After the copy is performed, LinkedList A and B should have identical but independent contents.
c) Write a function `int count(LinkedList &list, string name)` that counts the total number of Nodes in list that have `node->name == name`.

d) Write a function `bool insertBefore(LinkedList &list, Node* node, string name)` that inserts `Node* node` immediately before the `Node*` containing `name`, and return true. If no such name is found, then no insertion occurs, and false is returned.