CSE 5311 Lab Assignment 1

Due July 14

Goals:

1. Understanding of open address hashing.

2. Understanding of Brent's reorganization technique.

Requirements:

1. Write (and test) a program to evaluate Brent's rehash method. You should compare simple double hashing to double hashing with Brent's method. You should compare insertion and search performance on tables with at least 20,000 elements. You may use a random number generator to obtain the keys

2. Submit a printed listing of your entire program.

3. Submit printed listings of the execution traces from exercising your program on at least 3 different cases. You do not need to show each insertion.

4. Write a 1-page report summarizing what you learned from your executions.

Getting Started:

1. http://reptar.uta.edu/NOTES5311/cse5311.html provides web access to Pascal and C code for Brent's method. The C code is listed at the end of this handout.

2. The size of your hash tables should be a prime number. Otherwise, you will risk non-termination.

3. Measuring the performance at 5% intervals of the load factor will be sufficient.

4. Even though the provided code computes the expected number of probes by searching for each key, your program should maintain this information incrementally. In other words, the program will only do insertions, never searches.

```
#include <stdio.h>

#define TABSIZE 7
#define TABSIZELESS1 (TABSIZE - 1)

int hash[TABSIZE];
int n;

int hashfunction(int key)
{
return (key % TABSIZE);
}

int increment(int key)
{
int work;

work = (key / TABSIZE) % TABSIZE;
if (!work)
  work = 1;
return work;
}

void initTable()
{
int i;
n = 0;
for (i=0;i<TABSIZE;i++)
  hash[i] = (-1);
}
```

```
void insert (int key, int r[])
{
int i, ii, inc, init, j, jj;

init = hashfunction(key);
inc = increment(key);
for (i=0;i<=TABSIZE;i++)
{
  printf("trying to add just %d to total of probe lengths\n",i+1);
  for (j=i;j>=0;j--)
  {
    jj = (init + inc * j) % TABSIZE;
    ii = (jj + increment(r[jj]) * (i - j)) % TABSIZE;
    printf("i=%d j=%d jj=%d ii=%d\n",i,j,jj,ii);
    if (r[ii] == (-1))
    {
      r[ii] = r[jj];
      r[jj] = key;
      n++;
      return;
    }
  }
}
}

void doInserts()
{
int key;
int i, j, k;
int probelengthtotal;
for (k=1;k<=TABSIZE;k++)
{
  scanf("%d",&key);
  insert(key, hash);
  printf("hash=%d increment=%d\n",hashfunction(key),increment(key));
  for (i=0;i<TABSIZE;i++)
    printf("%d %d\n",i, hash[i]);
  probelengthtotal = 0;
  for (i=0;i<=TABSIZELESS1;i++)
    if (hash[i] > (-1))
    {
      j = hashfunction(hash[i]);
      probelengthtotal++;
      while (i != j)
      {
        j = (j + increment(hash[i])) % TABSIZE;
        probelengthtotal++;
      }
    }
  printf("probelengthtotal=%d expected=%f\n",probelengthtotal,
    ((float)probelengthtotal)/n);
}
}

main()
{
initTable();
doInserts();
}
```