Code for dynamic allocation of hash table of size m x m

```
typedef struct _wordR {
  char word[100];
  double x, y;
} wordR;

typedef struct _node{
  wordR w;
  struct _node *next;
} node, *nodePointer;


typedef nodePointer **hashTable;

int i, j, m;
hashTable H;

/*
  Allocating space for the hash table H and initializing the
  head pointers of the linked lists (each element of the
  hash table)
*/

   /*
      First allocate the row pointers. You have to set m to the
      proper size before this
   */
   H = (nodePointer **)malloc(m * sizeof(nodePointer *));

   /*
     Now allocate space for each row and initialize all
      pointers in the row to null
   */
   for (i=0; i<m; i++) {
      H[i] = (nodePointer *)malloc(m * sizeof(nodePointer));
      for (j=0; j<m; ++j) H[i][j] = NULL;
   }

/*
   Now you have the hash table H all set up. H[i][j] will contain
   the head pointer of the chained linked list storing all elements
   that hash to that slot.
   You should now use only H[i][j] in the rest of your program,
   do not try to use any pointer notation to access an element of H
*/
```

Code for file read and write

The code reads the tuple <word, x, y> from the input.txt (which is given in the format mentioned in the assignment) and prints it out in output.txt.

For your assignment, use the input part of the code to read from the file. For each <word, x,y> read, insert it in the hast table. Use the output part of the code to print appropriate elements to the output file (you wil have to change the code somewhat, this just shows how to write to a file).

```
FILE *inpf, *outf;
int n, i;
char tempW[100];
double tempX, tempY;


inpf = fopen(“input.txt”, “r”);
if (inpf == NULL) {
      printf(“Error opening input file input.txt\n”);
      return (-1);
}

outf = fopen(“output.txt”, “w”);
if (outf == NULL) {
      printf(“Error creating output file output.txt\n”);
      return (-1);
}

/* Read no. of elements from input file */
fscanf(inpf, “%d”, &n);

/*Write number of elements in out[put file */
fprintf(outf, “%d”, n);

/*
   Read each element from the input file and
   write it to the output file
*/
for (i=0; i<n; i++) {
      fscanf(inpf, “%s%lf%lf”, tempW, &tempX, &tempY);
      fprintf(outf, “%s  %lf  %lf\n”, tempW, tempX, tempY);
}

fclose(inpf);
fclose(outf);
```