Linked List - Inserting and deleting nodes

Name: Shreya Mamadapur Student ID: C0774035
Instructor: Takis Zourntos

To insert a new node, create a new node using the createNode() function and set it's values using setPaylode() function. To insert, we need to find out the node where we should insert it. This is done by passing the key at which point the new node has to be inserted. When this node is found, we take the node before this and make its pNext = the new node, also make the new node's pNext to the actual node with the matching key. This will insert the new node before the matching node.
To Delete a node, we have to first match the node with the key, and when this is found, we have to take the previous node to this and make it's pNext= the node after the matched node. This will essentially bypass the node to be deleted and as a final step, we have to free the memory used by the node by using the free() function.

Source Code

```
/*
 * ll.h
 *
 *  Created on: Jun. 29, 2020
 *      Author: takis
 */

#ifndef DSTRUCTS_LL_H_
#define DSTRUCTS_LL_H_

#include <stdlib.h>

/*************************************************************
 * NODE STRUCT GOES HERE
 *************************************************************/

// nodes can be referenced by a KEY, of this type
typedef unsigned int data_key_t;

// nodes contain a PAYLOAD consisting of various elements and the key
struct data_struct
{
      double X;
      double Y;
      data_key_t key;
};
typedef struct data_struct data_t;
```

Linked List - Inserting and deleting nodes

```c
// actual node struct/typedef
struct linkedList
{
      data_t payload;
      struct linkedList *pNext; // recursively defined "next" pointer
};
typedef struct linkedList ll_t;

/************************************************************
 * USEFUL LINKED LIST FUNCTIONS
 ************************************************************/
/*
 * setPayload():
 *
 *         for the node pointed to by the first argument, this function
 *         sets the value of the node's payload to the function's
 *         second argument
 *
 */
void setPayload(ll_t*, data_t);


/*
 * createNode():
 *
 *         creates a node of type ll_t from the heap, and returns
 *         a pointer to this newly created node; sets the node's own
 *       pNext pointer to NULL;
 *
 */
ll_t* createNode(void);

/*
 * addNode():
 *
 *         adds a new node (with payload given by second argument) to the
 *         bottom/back of the list referenced by the pointer, head;
 *         if head==NULL, a new list is created, and the new head pointer
 *         is returned;
 *
 */
```

Linked List - Inserting and deleting nodes

```
ll_t* addNode(ll_t*, data_t);

/*
 * insertNode():
 *
 *          for a list with head pointer given by the first argument, this
 *          function inserts a new node with payload given by the second
argument
 *          at the point just BEFORE the node whose key matches the third
 *          argument; if head pointer returns NULL, a new list is created,
and
 *          the new head pointer is returned;
 *
 */
ll_t* insertNode(ll_t*, data_t payload, data_key_t insertionPoint);

/*
 * deleteNode():
 *
 *          for a list with head pointer given by the first argument, this
 *          function deletes the node whose key data matches the second
argument;
 *          if head pointer returns NULL, an error is generated, indicating
that
 *          the node is not found;
 *
 */
void deleteNode(ll_t*, data_key_t);


#endif /* DSTRUCTS_LL_H_ */




/*
 * ll.c
 *
 *  Created on: Jun. 29, 2020
 *      Author: takis
 *      Revised on: July 19, 2020
 *      Revised by: Shreya
 */
```

Linked List - Inserting and deleting nodes

```c
#include <stdlib.h>
#include "ll.h"

/*
 * setPayload():
 */
void setPayload(ll_t* node, data_t payload)
{
      node->payload.X = payload.X;
      node->payload.Y = payload.Y;
      node->payload.key = payload.key;
}

/*
 * createNode():
 *
 */
ll_t* createNode(void)
{
      /* create a pointer for the new node */
      ll_t *node;

      /* allocate the node from heap */
      node = (ll_t*) malloc(sizeof(struct linkedList));

      /* make next point to NULL */
      node->pNext = NULL; //

      /* return the pointer to the new node */
      return node;
}

/*
 * addNode():
 *
 */
ll_t* addNode(ll_t *pHead, data_t payload)
{
      /* create two node pointers */
      ll_t *pNode;
      ll_t *pW;
```

Linked List - Inserting and deleting nodes

```
        /* prepare the new node to be added */
        pNode = createNode();
        setPayload(pNode, payload); /* set the new element's data field to 
value */

        if (pHead == NULL)
        {
                pHead = pNode; /* if the linked list has no nodes to begin with 
*/
        }
        else
        {
                /* search through list until tail node is found */
                pW = pHead;
                while ((pW->pNext) != NULL)
                {
                        pW = pW->pNext;
                }
                /* set the pointer from NULL to temp */
                pW->pNext = pNode;
        }
        return pHead;
}

/*
 * insertNode():
 *
 */
ll_t* insertNode(ll_t *pHead, data_t payload, data_key_t insertionPoint)
{
/*
 * This funciton creates a new node and places it before the node with 
data_key= insertionPoint
 */
        ll_t *newNode = createNode();//Create new node
        setPayload(newNode, payload);//Set the payload or the data in the 
node

        ll_t *prv = pHead;//Previous node to current node
        ll_t *pw = pHead;//Current node
```

Linked List - Inserting and deleting nodes

```
	while((pw->pNext) != NULL)//Search till the end of the ll
	{
		if(pw->payload.key == insertionPoint)
//If the key matches, we have to insert the new node here
		{
			prv->pNext=newNode;
// If the key matches then the new node should be inserted here. The
//previous node of the current node should point to the new node
			newNode->pNext = pw;
// The next node of the new node should point to the current node
			break;
		}
		prv = pw;//Previous node becomes the current node
		pw = pw->pNext;// Current node becomes next node

	}

	//return newNode;
	return pHead;
}

/*
 * deleteNode():
 *
 */
void deleteNode(ll_t *pHead, data_key_t nodeToDeleteKey)
{
	ll_t *prv = pHead;
	ll_t *pw = pHead;

	while((pw->pNext) != NULL)//Search till the end of the ll
	{
		if(pw->payload.key == nodeToDeleteKey) //If the key matches
		{
			prv->pNext=pw->pNext;
// If the key matches then remove the present node by equating the pNext of
the previous node to the pNext of the current node
			free(pw);
// Free the memory created by malloc() function in the CreateNode()
			break; //exit the loop if deleted
		}
		prv = pw;//Previous node becomes the current node
```

```
            pw = pw->pNext;// Current node becomes next node

      }

      return;
}

/*
 * test_ll.c
 *
 *
 * To use this code, enter the following at the command prompt ($):
 *
 * $ cat ../../data/data_text.txt | ./linked_list_lib | more
 *
 *
 *  Created on: Jun. 29, 2020
 *      Author: takis
 *      Revised by: Shreya
 */

#include <stdlib.h>
#include <stdio.h>
#include "ll.h"

int main(void)
{
      ll_t *pLLHead=NULL; // pointer to list, must be initialized to NULL
      data_t token;

      // create the linked list from standard input;
      // user indicates end of data by entering "9999"
      // for X, Y and key values.
      printf("\nLoading data...\n");
      scanf("%lf %lf %lu", &token.X, &token.Y, &token.key);
      pLLHead = addNode(pLLHead, token);
      while (token.X != 9999 && token.Y != 9999 && token.key != 9999)
      {
            scanf("%lf %lf %lu", &token.X, &token.Y, &token.key);
            addNode(pLLHead, token);
      }
      printf("     ... done.\n\n");
```

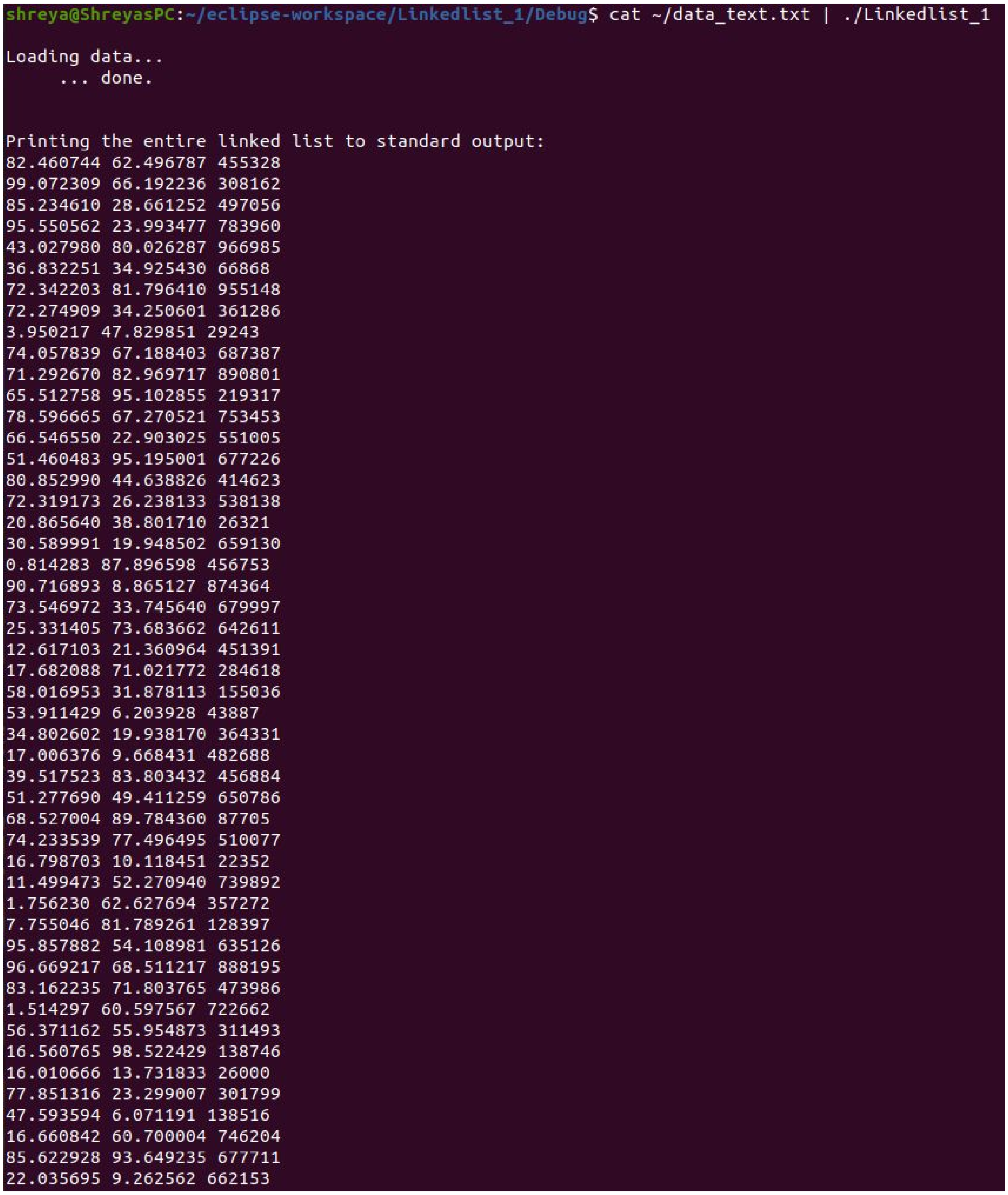
Linked List - Inserting and deleting nodes

```
	// send linked list to standard output
	printf("\nPrinting the entire linked list to standard output:\n");
	ll_t *pW = pLLHead;
	while (pW != NULL)
	{
		token.X = pW->payload.X;
		token.Y = pW->payload.Y;
		token.key = pW->payload.key;
		printf("%lf %lf %lu\n", token.X, token.Y, token.key);

		pW = pW->pNext;
	}

	printf("\n\nAdding another node before the node with key 290080 and
deleting the node with the key 166144\n");
	data_t newPaylod;
	newPaylod.X = 11;
	newPaylod.Y = 22;
	newPaylod.key = 112233;

	insertNode(pLLHead, newPaylod, 290080);//This will add a line with
key value of 112233 before the line with the key value of 290080
	deleteNode(pLLHead, 166144);//This will delete the line with the key
value of 166144

	printf("\nPrinting the New linked list to STDOUT:\n");
	pW = pLLHead;
	while (pW != NULL)
	{
		token.X = pW->payload.X;
		token.Y = pW->payload.Y;
		token.key = pW->payload.key;
		printf("%lf %lf %lu\n", token.X, token.Y, token.key);

		pW = pW->pNext;
	}


	// terminate
	printf("\n\n ...done!\n\n");
	return 0;
```

Linked List - Inserting and deleting nodes

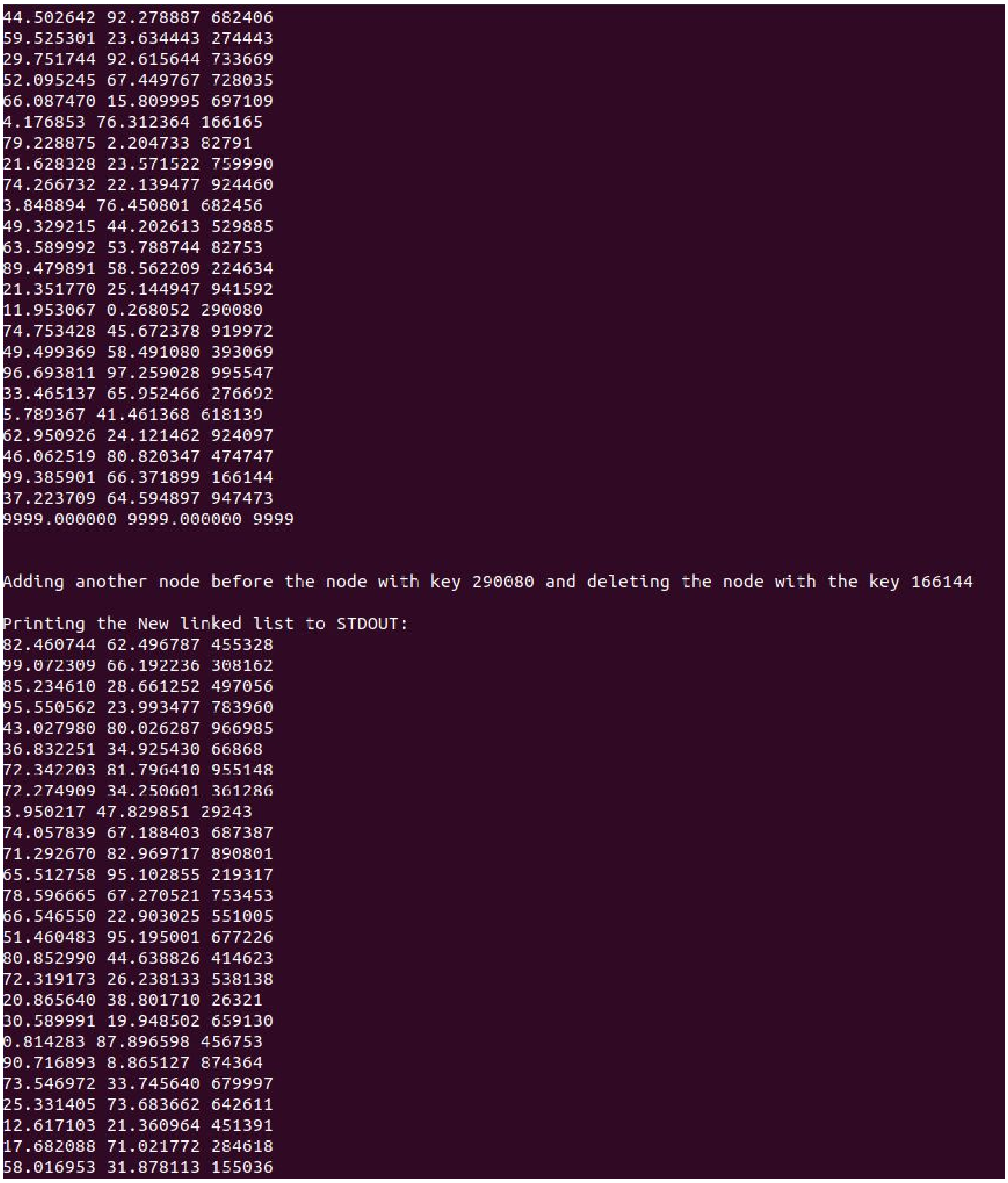
```
}
```

RESULT:



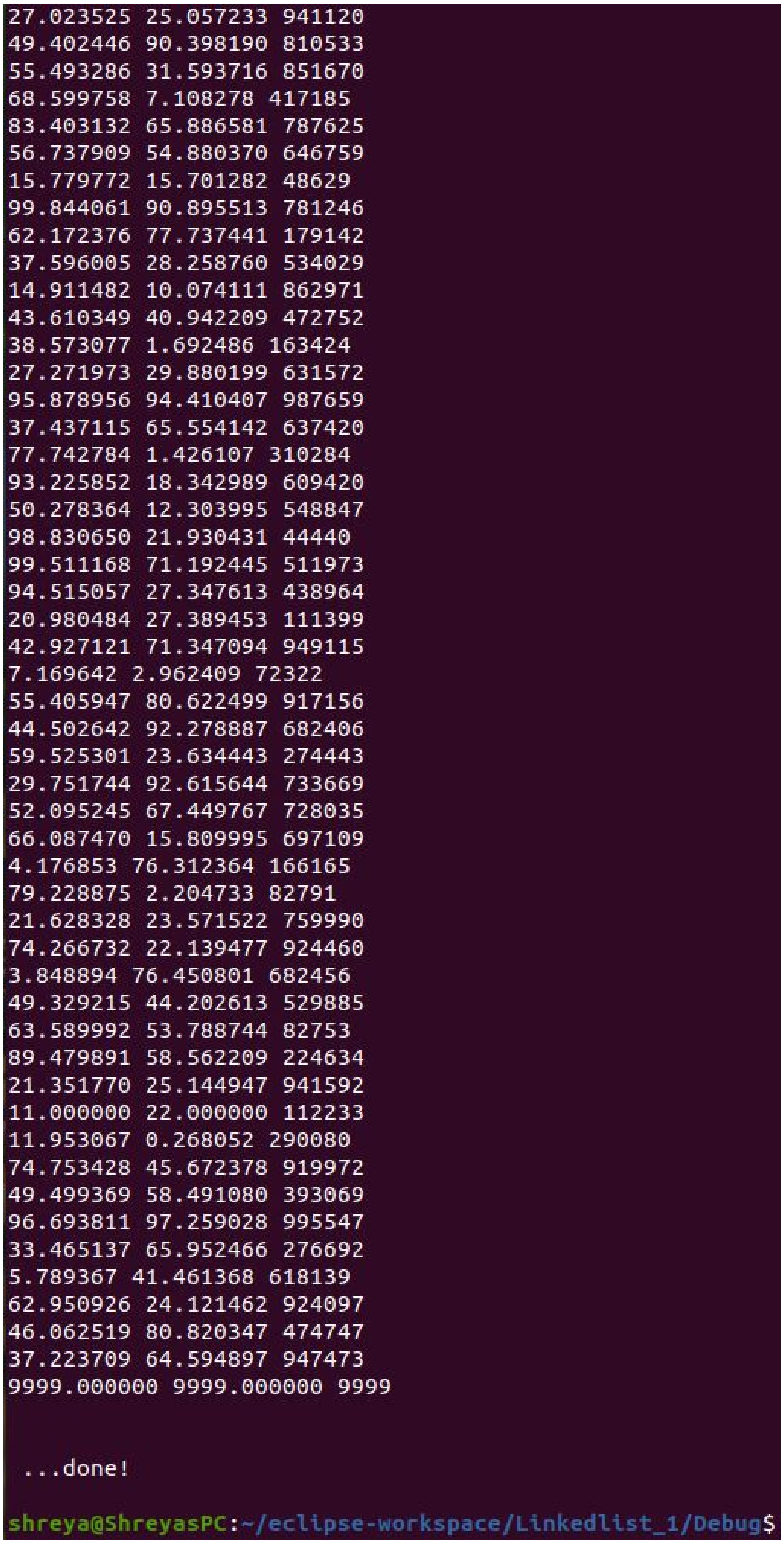
```
shreya@ShreyasPC:~/eclipse-workspace/Linkedlist_1/Debug$ cat ~/data_text.txt | ./Linkedlist_1

Loading data...
     ... done.


Printing the entire linked list to standard output:
82.460744 62.496787 455328
99.072309 66.192236 308162
85.234610 28.661252 497056
95.550562 23.993477 783960
43.027980 80.026287 966985
36.832251 34.925430 66868
72.342203 81.796410 955148
72.274909 34.250601 361286
3.950217 47.829851 29243
74.057839 67.188403 687387
71.292670 82.969717 890801
65.512758 95.102855 219317
78.596665 67.270521 753453
66.546550 22.903025 551005
51.460483 95.195001 677226
80.852990 44.638826 414623
72.319173 26.238133 538138
20.865640 38.801710 26321
30.589991 19.948502 659130
0.814283 87.896598 456753
90.716893 8.865127 874364
73.546972 33.745640 679997
25.331405 73.683662 642611
12.617103 21.360964 451391
17.682088 71.021772 284618
58.016953 31.878113 155036
53.911429 6.203928 43887
34.802602 19.938170 364331
17.006376 9.668431 482688
39.517523 83.803432 456884
51.277690 49.411259 650786
68.527004 89.784360 87705
74.233539 77.496495 510077
16.798703 10.118451 22352
11.499473 52.270940 739892
1.756230 62.627694 357272
7.755046 81.789261 128397
95.857882 54.108981 635126
96.669217 68.511217 888195
83.162235 71.803765 473986
1.514297 60.597567 722662
56.371162 55.954873 311493
16.560765 98.522429 138746
16.010666 13.731833 26000
77.851316 23.299007 301799
47.593594 6.071191 138516
16.660842 60.700004 746204
85.622928 93.649235 677711
22.035695 9.262562 662153
```

Linked List - Inserting and deleting nodes



```
44.502642 92.278887 682406
59.525301 23.634443 274443
29.751744 92.615644 733669
52.095245 67.449767 728035
66.087470 15.809995 697109
4.176853 76.312364 166165
79.228875 2.204733 82791
21.628328 23.571522 759990
74.266732 22.139477 924460
3.848894 76.450801 682456
49.329215 44.202613 529885
63.589992 53.788744 82753
89.479891 58.562209 224634
21.351770 25.144947 941592
11.953067 0.268052 290080
74.753428 45.672378 919972
49.499369 58.491080 393069
96.693811 97.259028 995547
33.465137 65.952466 276692
5.789367 41.461368 618139
62.950926 24.121462 924097
46.062519 80.820347 474747
99.385901 66.371899 166144
37.223709 64.594897 947473
9999.000000 9999.000000 9999


Adding another node before the node with key 290080 and deleting the node with the key 166144

Printing the New linked list to STDOUT:
82.460744 62.496787 455328
99.072309 66.192236 308162
85.234610 28.661252 497056
95.550562 23.993477 783960
43.027980 80.026287 966985
36.832251 34.925430 66868
72.342203 81.796410 955148
72.274909 34.250601 361286
3.950217 47.829851 29243
74.057839 67.188403 687387
71.292670 82.969717 890801
65.512758 95.102855 219317
78.596665 67.270521 753453
66.546550 22.903025 551005
51.460483 95.195001 677226
80.852990 44.638826 414623
72.319173 26.238133 538138
20.865640 38.801710 26321
30.589991 19.948502 659130
0.814283 87.896598 456753
90.716893 8.865127 874364
73.546972 33.745640 679997
25.331405 73.683662 642611
12.617103 21.360964 451391
17.682088 71.021772 284618
58.016953 31.878113 155036
```

Linked List - Inserting and deleting nodes



```
27.023525 25.057233 941120
49.402446 90.398190 810533
55.493286 31.593716 851670
68.599758 7.108278 417185
83.403132 65.886581 787625
56.737909 54.880370 646759
15.779772 15.701282 48629
99.844061 90.895513 781246
62.172376 77.737441 179142
37.596005 28.258760 534029
14.911482 10.074111 862971
43.610349 40.942209 472752
38.573077 1.692486 163424
27.271973 29.880199 631572
95.878956 94.410407 987659
37.437115 65.554142 637420
77.742784 1.426107 310284
93.225852 18.342989 609420
50.278364 12.303995 548847
98.830650 21.930431 44440
99.511168 71.192445 511973
94.515057 27.347613 438964
20.980484 27.389453 111399
42.927121 71.347094 949115
7.169642 2.962409 72322
55.405947 80.622499 917156
44.502642 92.278887 682406
59.525301 23.634443 274443
29.751744 92.615644 733669
52.095245 67.449767 728035
66.087470 15.809995 697109
4.176853 76.312364 166165
79.228875 2.204733 82791
21.628328 23.571522 759990
74.266732 22.139477 924460
3.848894 76.450801 682456
49.329215 44.202613 529885
63.589992 53.788744 82753
89.479891 58.562209 224634
21.351770 25.144947 941592
11.000000 22.000000 112233
11.953067 0.268052 290080
74.753428 45.672378 919972
49.499369 58.491080 393069
96.693811 97.259028 995547
33.465137 65.952466 276692
5.789367 41.461368 618139
62.950926 24.121462 924097
46.062519 80.820347 474747
37.223709 64.594897 947473
9999.000000 9999.000000 9999


 ...done!

shreya@ShreyasPC:~/eclipse-workspace/Linkedlist_1/Debug$
```

Linked List - Inserting and deleting nodes

As we see in the above images, a key called 112233 is inserted and the key 166144 is deleted.