CPE112: Linked List

Data Structure - Lab 3

Linked List - Display the List

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
//display the list
void printList() {
   struct node *ptr = head;
   printf("\n[ ");

   //start from the beginning
   while(ptr != NULL) {
      printf("(%d,%d) ",ptr->key,ptr->data);
      ptr = ptr->next;
   }

   printf(" ]");
}
```

Linked List - Insert First

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
//insert link at the first location
void insertFirst(int key, int data) {
   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct node));

   link->key = key;
   link->data = data;

   //point it to old first node
   link->next = head;

   //point first to new first node
   head = link;
}
```

Linked List - main function

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
void main() {
   insertFirst(1,10);
   insertFirst(2,20);
   insertFirst(3,30);
   insertFirst(4,1);
   insertFirst(5,40);
   insertFirst(6,56);

   printf("Original List: ");

   //print list
   printList();
}
```

OUTPUT

```
/tmp/HaAzoRmVZg.o
Original List:
[ (6,56) (5,40) (4,1) (3,30) (2,20) (1,10)  ]
```

Linked List - Delete First

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
//delete first item
struct node* deleteFirst() {

   //save reference to first link
   struct node *tempLink = head;

   //mark next to first link as first
   head = head->next;

   //return the deleted link
   return tempLink;
}

```

Linked List - isEmpty and length()

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
//is list empty
bool isEmpty() {
   return head == NULL;
}

int length() {
   int length = 0;
   struct node *current;

   for(current = head; current != NULL; current = current->next) {
      length++;
   }

   return length;
}
```

Linked List - main function

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
void main() {
   insertFirst(1,10);
   insertFirst(2,20);
   insertFirst(3,30);
   insertFirst(4,1);
   insertFirst(5,40);
   insertFirst(6,56);

   printf("Original List: ");

   //print list
   printList();

   while(!isEmpty()) {
      struct node *temp = deleteFirst();
      printf("\nDeleted value:");
      printf("(%d,%d) ",temp->key,temp->data);
   }

   printf("\nList after deleting all items: ");
   printList();
}
```

Linked List - main function

```c
//delete first item
struct node* deleteFirst() {

   //save reference to first link
   struct node *tempLink = head;

   //mark next to first link as first
   head = head->next;

   //return the deleted link
   return tempLink;
}
```

```c
   while(!isEmpty()) {
      struct node *temp = deleteFirst();
      printf("\nDeleted value:");
      printf("(%d,%d) ",temp->key,temp->data);
   }

   printf("\nList after deleting all items: ");
   printList();
}
```

OUTPUT

```
Original List:
[ (6,56) (5,40) (4,1) (3,30) (2,20) (1,10)  ]
Deleted value:(6,56)
Deleted value:(5,40)
Deleted value:(4,1)
Deleted value:(3,30)
Deleted value:(2,20)
Deleted value:(1,10)
List after deleting all items:
[  ]
```

Linked List - Find

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
//find a link with given key
struct node* find(int key) {

   //start from the first link
   struct node* current = head;

   //if list is empty
   if(head == NULL) {
      return NULL;
   }

   //navigate through list
   while(current->key != key) {

      //if it is last node
      if(current->next == NULL) {
         return NULL;
      } else {
         //go to next link
         current = current->next;
      }
   }

   //if data found, return the current Link
   return current;
}
```

Linked List - Delete a Link with a Given Key

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
//delete a link with given key
struct node* delete(int key) {

   //start from the first link
   struct node* current = head;
   struct node* previous = NULL;

   //if list is empty
   if(head == NULL) {
      return NULL;
   }

   //navigate through list
   while(current->key != key) {

      //if it is last node
      if(current->next == NULL) {
         return NULL;
      } else {
         //store reference to current link
         previous = current;
         //move to next link
         current = current->next;
      }
   }

   //found a match, update the link
   if(current == head) {
      //change first to point to next link
      head = head->next;
   } else {
      //bypass the current link
      previous->next = current->next;
   }

   return current;
}
```

Linked List - Delete a Link with a Given Key

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;
```

```c
struct node *foundLink = find(4);

if(foundLink != NULL) {
   printf("Element found: ");
   printf("(%d,%d) ",foundLink->key,foundLink->data);
   printf("\n");
} else {
   printf("Element not found.");
}

delete(4);
printf("List after deleting an item: ");
printList();
printf("\n");
foundLink = find(4);

if(foundLink != NULL) {
   printf("Element found: ");
   printf("(%d,%d) ",foundLink->key,foundLink->data);
   printf("\n");
} else {
   printf("Element not found.");
}
```

Linked List - Delete a Link with a Given Key

```
struct node *foundLink = find(4);

if(foundLink != NULL) {
   printf("Element found: ");
   printf("(%d,%d) ",foundLink->key,foundLink->data);
   printf("\n");
} else {
   printf("Element not found.");
}

delete(4);
printf("List after deleting an item: ");
printList();
printf("\n");
foundLink = find(4);

if(foundLink != NULL) {
   printf("Element found: ");
   printf("(%d,%d) ",foundLink->key,foundLink->data);
   printf("\n");
} else {
   printf("Element not found.");
}
```

OUTPUT

```
Element found: (4,1)
List after deleting an item:
[ (6,56) (5,40) (3,30) (2,20) (1,10)  ]
Element not found.
```