Singly Linked List

Operations

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING,
KANCHEEPURAM

Dr. Preeth R

Assistant Professor

Department of Computer Sc. and Engg.

Insert at beginning(Passing *head* pointer)

```
struct Node *insertAtBeginning(struct Node* head, int data)
{
        printf("Header address=%p\t",head);
       struct Node* temp= (struct Node*)malloc(sizeof(struct Node));
         temp->data = data;
         temp->next = NULL;
    if (head==NULL)
{
         head =temp;
       return head;
}

    else {



          temp->next= head;
          head =temp;
        return head;

    }
}
```

Insert at end

```
struct Node *insertatEnd(struct Node *head, int data)
{
    struct Node *last = head;
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

    new_node->data = data;
    new_node->next = NULL;
    if (head == NULL)
    {
        head = new_node;
        return head ;
    }
    while (last->next != NULL)
        last = last->next;
    last->next = new_node;
    return head ;
}
```

Insert at Position

```
struct Node *insertAtPosition(struct Node* head, int pos, int data)
{

        struct Node *curr=head;
         struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
          new_node->data = data;
          new_node->next=NULL;
         int i=1;
        while (i != pos-1)
         {
              curr=curr->next;
              i=i+1;
              printf("Hi");
              if (curr->next==NULL)
              {
                   printf("Sorry");
                   return 0;
              }
         }

         new_node->next=curr->next;
         curr->next=new_node;
              head;
    }
```

Print/Traverse

```
void printList(struct Node* head)
{
   // printf("Head address=%p\t",head);
    while (head != NULL) {
        printf("\n");
        printf("%d\t%p ",head->data,head->next);
        head = head->next;
    }
    printf("\n");
}
```

Delete at start

```
struct Node *deleteStart (struct Node *head)
{
  struct Node *temp = head;

  // if there are no nodes in Linked List can't delete
  if (head == NULL)
    {
      printf ("Linked List Empty, nothing to delete");
      return head ;
    }
  // move head to next node
   head = head->next;
   free (temp);
   temp=NULL;
   return head;
}
```

Delete at Position

```
struct Node *deletePos(struct Node *head, int pos)
{
    struct Node *temp =head;
     struct Node *prev = head;
     for (int i = 0; i < pos; i++) {
          if (i == 0 && pos == 1) {
               head = head->next;
               free(temp);
               return head;
          }
          else {
               if (i == pos - 1 && temp!=NULL)
                {
                    prev->next = temp->next;
                    free(temp);
               }
               else {
                    prev = temp;
                     if (prev == NULL) {
                         break;
                          }
                    temp = temp->next;
               }
          }
     }
}
```

Main Function

```
int main()
{
     struct Node* head = NULL;
head= insertAtBeginning(head,5);
head= insertAtBeginning(head,6);
  //head= insertAtBeginning(head,  10);
   //head= insertAtBeginning(head,  15);
   head=insertatEnd(head,10);
   head=insertatEnd(head,15);
   head=insertatEnd(head,20);
   head=insertatEnd(head,25);

    printList(head);
    insertAtPosition(head,6,35);
    printf("Linked list after insertion: ");
    printList(head);

   //  head= deleteStart(head);
 //printf("Linked list after deletion: ");
   // printList(head);
    return 0;
}
```

Take Away Tasks

- DeleteatEnd()
- Deleteatposition()
- Incorporation of identifying invalid position value in insertAtPosition().