বিভিন্ন তথ্য রাখার জন্য বিভিন্ন ধরনের লিস্ট তৈরি করা হয়। লিস্টের একটি পরিচিত রূপ হল অ্যারে। অতগুলো তথ্য রাখতে হবে সেই অনুযায়ী একটা অ্যারে ডিক্লেয়ার করে সেখানে তথ্যকে লিস্ট আকারে রাখা যায় এবং পরবর্তিতে বিভিন্ন কাজ করা যায়। কিন্তু অ্যারেতে বেশ কিছু সমস্যা রয়েছে। যখন অ্যারে ডিক্লেয়ার করা হয় তখন সেখানে বলে দিতে হয় কতগুলো এলিমেন্ট থাকবে। যেহেতু অধিকাংশ ক্ষেত্রেই আমরা বলতে পারি না কতগুলো এলিমেন্ট লিস্টে থাকতে পারে তাই আনুমানিক আকারের একটা লিস্ট তৈরি করতে হয়। এতে করে অনেকগুলো এলিমেন্ট ডিক্লেয়ার করার ফলে অনেক মেমরি অপচয় হয়।

এছাড়া আরও একটি সমস্যা রয়েছে। অ্যারে মেমরিতে অবিচ্ছিন্নভাবে থাকে। অর্থাৎ একটার পর একটা এলিমেন্ট একের পর এক মেমরি ব্লকে থাকে। এর মাঝে আর কিছু থাকে না। আমাদের কাছে অনেকগুলো এলিমেন্টের একটা লিস্ট আছে। এখন যদি নতুন একটা এলিমেন্টকে সেই লিস্টের মাঝে কোথাও অন্তর্ভুক্ত করতে হয় তাহলে যে অবস্থানে এলিমেন্টটিকে রাখতে হবে সেখান থেকে তারপরের সবগুলো এলিমেন্টকে একটা একটা করে সরিয়ে পিছে নিতে হবে এবং মাঝে একটা জায়গা ফাঁকা করতে হবে। তারপর নতুন এলিমেন্ট রাখতে হবে। এটা অনেক সময়সাপেক্ষ একটা প্রক্রিয়া। এছাড়া যদি এমন হয় যে আবার কোন একটা এলিমেন্ট ডিলিট করতে হবে, তাহলে আবার পিছের সব এলিমেন্টকে এক এক করে সামনে নিয়ে আসতে হবে। আর এই নতুন এলিমেন্ট যোগ করা বা কোন এলিমেন্ট ডিলিট করার প্রক্রিয়া যদি বারবার করতে হয় তাহলে প্রোগ্রামটি অনেক সময় নিবে কাজগুলো করতে।

এইসব সমস্যা দূর করার জন্যই লিংক লিস্ট। এই লিস্টের বৈশিষ্ট্য হল এখানে এলিমেন্টগুলো মেমরিতে অবিচ্ছিন্নভাবে থাকে না। বরং বিচ্ছিন্নভাবে বিভিন্ন জায়গায় ছড়িয়ে ছিটিয়ে থাকে। অ্যারেতে যেহেতু একের পর এক থাকে তাই সেখানে প্রথম এলিমেন্টের এড্রেস জানলে সেই এড্রেসের সাথে এক করে যোগ করে এগিয়ে যেতে থাকলেই পরের এলিমেন্টগুলো পাওয়া যেত। কিন্তু এখানে তো একের পর এক থাকবে না। তাহলে আমরা প্রথম এলিমেন্টের পরের এলিমেন্ট পাব কিভাবে? সেইজন্য আমাদেরকে প্রতিটা এলিমেন্টের সাথে পরবর্তি এলিমেন্টটি যেখানে আছে তার এড্রেস রাখতে হয়। ফলে আমরা যখন একটা এলিমেন্টে যাব তখন এর পরবর্তি এলিমেন্টের এড্রেস তার সাথে থাকায় সেটা দিয়ে আমরা পরের এলিমেন্টে চলে যেতে পারি। আবার সেখানে তার পরের এলিমেন্টের এড্রেস থাকায় তার পরের এলিমেন্টে চলে যেতে পারি। এভাবে একটা চেইন তৈরি হয়। আর এই

তাহলে এটুকু বোঝা গেল যে অ্যারেতে যেমন লিস্টের একেকটা এলিমেন্টে শুধু ঐ এলিমেন্টটাই থাকে, লিংক লিস্টে তেমনটি থাকে না। লিংক লিস্টে প্রতিটা এলিমেন্টের সাথে সাথে তার পরবর্তি এলিমেন্টের এড্রেস থাকে। তাহলে লিস্টের শুরু কোথায় সেটা জানবো কিভাবে? সেটা জানার জন্য আমরা যখন লিস্ট তৈরি করবো তখন প্রথম এলিমেন্টের এড্রেসটা আমাদের কাছে একটা পয়েন্টারে রেখে দিব। এটা অনেক

গুরুত্বপূর্ণ ব্যাপার। প্রথম এলিমেন্টের এড্রেস না থাকলে কিন্তু লিস্টের কিছুই পাওয়া যাবে না। হারিয়ে যাবে। কারণ অন্য সব এলিমেন্টের এড্রেস তার আগের এলিমেন্টের সাথে আছে। আর লিস্ট শেষ হল কোথায় সেটা জানবো/বুঝবো কিভাবে? শেষ যে এলিমেন্ট থাকবে তার সাথে তার পরবর্তি এলিমেন্টের এড্রেস যে থাকার কথা, সেই এড্রেসটা হবে NULL। অতএব লিস্টের একেকটা এলিমেন্ট ধরে এগোতে থাকলে যখনই আমরা NULL পয়েন্টার পেয়ে যাব তখ

এখন আমরা তাহলে লিংক লিস্টের মূল বৈশিষ্ট্য জানলাম। আসল ব্যাপারটাই এখনও পরিষ্কার করা হয় নি। অ্যারেতে যে সমস্যা ছিল যে কোন একটা এলিমেন্ট যদি এর মাঝে বসাতে হয় তাহলে পরের সবগুলাকে সরাতে হয়, লিংক লিস্টে সেটি না করেই কিভাবে বসানো যায়? সহজ উত্তর। যে দুইটা এলিমেন্টের মাঝে আমাকে নতুন এলিমেন্ট বসাতে হবে সেই দুইটা এলিমেন্ট আমরা খুঁজে বের করবো। তারপর আগের এলিমেন্টের যে স্থানে পরবর্তি এলিমেন্টের এড্রেস ছিল সেখানে নতুন এলিমেন্টের এড্রেস দিয়ে দিব। আর নতুন এলিমেন্টের সাথে পরবর্তি এলিমেন্টের এড্রেস রাখার যে ব্যবস্থা আছে সেখানে পরের যে এলিমেন্টটা আছে তার এড্রেস বসিয়ে দিব। অর্থাৎ যদি এমন হয় আমাকে x এবং y এর মাঝে a কে বসাতে হয় তাহলে x এর সাথে a এর এড্রেস রাখবো এবং a এর সাথে y এর এড্রেস রাখবো। তাহলে যখন আমরা x এ যাব তখন এরপরের এলিমেন্টে গেলে a পাব। কারণ a এর এড্রেস রাখা আছে। আবার a থেকে আমরা পরের এলিমেন্ট y এ যাব। কারণ a এর সাথে y এর এড্রেস আছে। তাহলে বস্তুত পক্ষে যেটা হল তা হল, এলিমেন্টগুলো মেমরিতে যেখানে যেই এড্রেসে আছে সেখানেই রয়ে যাচ্ছে। আমরা কেবল পরের এলিমেন্ট কোনটা হবে সেটার এড্রেস পালটিয়ে দিলেই লিংক লিস্টের ক্রম পালটিয়ে যাচ্ছে। ফলে অনেকগুলো এলিমেন্টকে এক এক করে কোথাও সরাতে হচ্ছে না।

এখন পর্যন্ত যা আলোচনা হল তাতে এটা বোঝা গেল যে এখানে একটা এলিমেন্ট একা থাকবে না। তার সাথে পরের এলিমেন্টের এড্রেস থাকবে। কিন্তু একসাথে একটা ডাটা টাইপে আমরা দুই ধরনের ডাটা রাখতে পারি না। সেটা করতে হলে আমাদের লাগবে স্ট্রাকচার। অতএব লিংক লিস্ট বুঝতে হলে আমাদেরকে স্ট্রাকচার সম্পর্কেও ভালো ধারনা রাখতে হবে। আমরা একেকটা এলিমেন্টকে বলব নোড। কারণ লিস্টে যে কেবল একটা করে এলিমেন্ট নিয়েই তৈরি হবে তা না। এমন যদি হয় ১০০ জন ছাত্রের তথ্য আছে। ছাত্রের নাম, রোল, ঠিকানা। যদি ১০০ জন ছাত্রের লিস্ট তৈরি করা হয় তবে সেখানে প্রতিটা নোড হবে একেকজন ছাত্রের জন্য আর সেখানে ৪ টা করে তথ্য থাকবে। নামের জন্য একটা স্ট্রিং, রোলের জন্য একটা int, ঠিকানার জন্য আরেকটা স্ট্রিং এবং পরবর্তি ছাত্রের তথ্য কোন এড্রেসে আছে সেটার পয়েন্টার। অতএব একটা লিস্ট তৈরি হয় কতগুলো নোড দিয়ে। নোডগুলো কিরকম হবে তা আমরা স্ট্রাকচার ডিক্লেয়ার করে বলে দিব। বোঝার সুবিধার জন্য এখানে কেবল কতগুলো সংখ্যার লিস্টের ব্যাপারে আলোচনা করা হল। এতে একটা নোডে একটা int থাকবে, যেই সংখ্যা আমরা রাখতে চাই। আর থাকবে পরের নোডের এড্রেস। তাহলে একটা স্ট্রাকচার ডিফাইন করা যাক।

```
struct aNode {
  int nodeValue; // the value of the node
  struct aNode *nextNode; // the address of the next node
};
```

এখানে আমরা aNode নামে একটা নতুন ডাটা টাইপ তৈরি করলাম। প্রতিটা নোড এই টাইপের হবে। আগেই বলা হয়েছে আমরা এই লিস্ট ব্যবহার করতে চাইলে তার প্রথম নোডের এড্রেস আমাদের রাখতে হবে। সেই জন্য একটা পয়েন্টার ডিক্লেয়ার করব।

```
typedef struct aNode Node;
Node *head;
```

প্রথমে আমরা struct aNode ডাটা টাইপের একটা প্রতিশব্দ তৈরি করলাম typedef ব্যবহার করে। কারণ আমাদেরকে অনেক বড় একটা কোডের মধ্যে বারবার struct aNode লিখতে হবে। সেখানে এই পুরো অংশটার জন্য একটা প্রতিশব্দ তৈরি করলাম Node নামে। ফলে যেখানেই Node লেখা থাকবে আমরা বুঝবো যে আসলে সেটা struct aNode লেখা। ব্যাপারটা বুঝতে অসুবিধা হলে এভাবে চিন্তা কর যে এরপর থেকে কোডের যেখানেই দেখবে Node লেখা, সেখানে তুমি struct aNode বসিয়ে নিবে। ভালোমত ধারনা পেতে চাইলে এই লেখাটা পড়তে পারো।

এরপর আমরা head নামে একটা পয়েন্টার ডিক্লেয়ার করলাম যার মধ্যে লিস্টের প্রথম নোডের এড্রেস থাকবে। এই head কে আমরা গ্লোবাল ভেরিয়েবল হিসেবে ডিক্লেয়ার করলাম কারণ বিভিন্ন ফাংশন একে ব্যবহার করবে। যখন আমরা main ফাংশনে কাজ শুরু করব তখন প্রথমেই head = NULL। এটা বোঝাবে যে এখনও লিস্ট তৈরি হয়নি। আমরা যখন head এ প্রবেশ করতে চাব তখন NULL থাকার কারণে কোথাও প্রবেশ করা যাবে না। ফলে আমরা বুঝতে পারব যে এখনও লিস্ট তৈরি হয়নি।

লিস্ট তৈরি এবং ব্যবহারে সাধারণভাবে তিনধরনের কাজ করা হয় -
১) নতুন নোড যোগ করা হয় (insert node)
২) কোন নোড ডিলিট করা হয় (delete node)
৩) সমগ্র লিস্ট প্রিন্ট করা হয় (print list)

এই তিনটা কাজের জন্য আমরা তিনটা ফাংশন লিখবো -
১) void insertNode(int insertValue)
২) void deleteNode(int deleteValue)
৩) void printList( )

প্রথমেই দেখা যাক insertNode() কিভাবে কাজ করে।

```
void insertNode( int insertValue )
{
  Node *temp;
```

```
  Node *current;
  temp = ( Node * ) malloc( sizeof( Node ) );
  temp->nodeValue = insertValue;
  temp->nextNode = NULL;
  if( !head ) {
    head = temp;
  }
  else if( insertValue <= head->nodeValue ) {
    temp->nextNode = head;
    head = temp;
  }
  else {
    for( current = head; current->nextNode; current = current->nextNode ) {
      if( insertValue > current->nodeValue && insertValue <= current->nextNode->nodeValue )
        break;
    }
    if( !( current->nextNode ) )
      current->nextNode = temp;
    else {
      temp->nextNode = current->nextNode;
      current->nextNode = temp;
    }
  }
  printf( "Value has been inserted successfully!\n" );
  printList();
  return;
}
```

৪ ও ৫ নাম্বার লাইনে দুইটা পয়েন্টার ডিক্লেয়ার করা হয়েছে। পয়েন্টার দুইটা Node এর এড্রেস রাখতে পারবে। এরপর আমরা malloc( ) ব্যবহার করে Node এর জন্য মেমরি এলোকেট করলাম। এটা আমরা যে নাম্বারটা লিস্টে রাখতে চাই তার জন্য তৈরি করা নতুন নোড। এখানে আমরা নাম্বারটাকে রাখব। এই যে মেমরি এলোকেট করলাম, যেখানে মেমরি এলোকেট করা হয়েছে সেই এড্রেসটা আমরা temp এর মধ্যে রাখলাম। এরপর আমরা যে লিখলাম temp->nodeValue = insertValue তার মানে হল, temp এর মধ্যে যে এড্রেস আছে, অর্থাৎ আমাদের নতুন তৈরিকৃত নোডের এড্রেস, সেই এড্রেসে যে স্ট্রাকচার আছে (একটু আগেই আমরা মেমরি এলোকেট করে তৈরি করেছি) তার nodeValue ভেরিয়েবলের মধ্যে আমরা আমাদের insertValue কে রাখলাম। আর ঐ একই স্ট্রাকচারের মধ্যে পরবর্তি নোডের এড্রেস রাখার যে পয়েন্টার তার মধ্যে NULL রাখলাম। কারণ এখনও আমরা জানি না এর পরে কি আর কিছু থাকবে নাকি।

এরপর যেটা করা হল তা হল, if(!head) বা যদি head এর ভেলু NULL হয় তাহলে আমরা জানি যে এখনও লিস্ট তৈরি হয়নি। অর্থাৎ এটাই লিস্টের প্রথম এবং শেষ, একমাত্র নোড হবে। তাই আমরা head এর মধ্যে temp এ থাকা এড্রেস, অর্থাৎ আমাদের নতুন নোডের এড্রেস রেখে দিলাম।

এখন যদি head এর ভেলু NULL না হত, মানে যদি লিস্টে আগে থেকেই কিছু না কিছু থাকত, তাহলে আমরা প্রথমে দেখতাম যে যেই নাম্বারটা লিস্টে রাখতে হবে সেটা কি head এ থাকা এড্রেসে থাকা স্ট্রাকচারে থাকা (এরপর থেকে এভাবে বলব "head এ থাকা ") নাম্বারের সমান কিংবা ছোট কিনা। কারণ আমরা এখানে একটা ordered লিংক লিস্ট তৈরি করছি। এখানে প্রতিটা তথ্য যখন ইনপুট দেওয়া হবে তখন তা সর্ট হয়ে লিস্টে জায়গা মত বসবে। তাই যদি head এর ভেলু থেকে ছোট বা সমান হয় তাহলে সেই নাম্বারকে আমরা head এর আগে বসাবো। আর এটাই হবে আমাদের নতুন head।

এই জন্য আমরা প্রথমে বর্তমান head এ যে এড্রেস আছে তাকে নতুন নোডের পরবর্তি নোডের এড্রেসের মধ্যে রেখে দিলাম। তারপর head পয়েন্টারের মধ্যে আমরা আমাদের নতুন সৃষ্ট নোডের এড্রেস রেখে দিলাম। তাহলেই নতুন নোডটা শুরুতে যোগ হয়ে গেল এবং সেখানে গেলে আমরা পরের নোডের এড্রেসে গেলে আগে যেটি head ছিল (এখন যেটি দ্বিতীয়) সেখানে যেতে পারব।

যদি ভেলুটি head এর ভেলু থেকে বড় হয়, তাহলে আমরা একটা লুপ চালাবো, head থেকে লুপ শুরু করবো এবং তা চলতে থাকবে যতক্ষণ না পর্যন্ত আমরা ভেলুটিকে বসানোর উপযুক্ত স্থান পাচ্ছি। অর্থাৎ যখন আমরা দেখবো যে লিস্টের এমন অবস্থানে চলে এসেছি যেখানে পূর্ববর্তি ভেলুটি নতুন ভেলুর চেয়ে ছোট এবং পরবর্তি ভেলুটি এর সমান বা বড় তখন আমরা সেখানে নতুন নোডটিকে বসাবো। যদি এমন অবস্থান না পাই, নতুন ভেলুটি যদি লিস্টের সব ভেলুর চেয়ে বড় হয় তাহলে তাকে লিস্টের একেবারে শেষে নিয়ে বসাবো। এখন দেখা যাক লুপটা কিভাবে কাজ করে।

লুপ চালানোর জন্য এখানে current নামক পয়েন্টারটা ব্যবহার করা হয়েছে। অ্যারেতে লুপ চালানোর সময় আমরা যেমন int i ব্যবহার করি (সাধারনত) তেমনি এখানে যেহেতু এড্রেস ধরে লুপ চালানো হবে তাই পয়েন্টার ব্যবহার করা হচ্ছে। শুরুতে আমরা current এর মধ্যে head এ থাকা এড্রেস রাখলাম। head থেকে লুপ শুরু হবে। এরপর আমরা দেখলাম current এর মধ্যে যে পরবর্তি নোডের এড্রেস আছে সেটা NULL কি না। যদি NULL হয় তার মানে আমরা লিস্টের শেষে চলে এসেছি। যদি তা না হয় তাহলে আমরা লুপের ভিতরে ঢুকলাম। এখন আমরা current এ থাকা ভেলু এবং current এর পরবর্তি নোডে থাকা ভেলুর সাথে নতুন ভেলু তুলনা করে দেখবো যে সেটা কি এদের মাঝে বসবে কি না। যদি বসে তার মানে আমরা অবস্থান পেয়ে গিয়েছি, তাই লুপ ব্রেক করব। আর যদি না বসে, তাহলে আমরা current এর মধ্যে বর্তমান current এ থাকা নোডের পরবর্তি নোডের এড্রেস দিয়ে দিব। এভাবে একটা একটা করে নোড আগাতে থাকবে এবং লুপ চলতে

যখন লুপ ব্রেক হয়ে আসলো তখন আমরা দেখবো সেটি কিভাবে ব্রেক হয়েছে। আমরা কি কোন অবস্থান পেয়েছি নাকি লিস্টের শেষে চলে এসেছি। যদি লিস্টের শেষে চলে আসি তাহলে current->nextNode এর মধ্যে থাকা এড্রেস NULL হবে। যদি সেটি হয় তাহলে

আমরা সেখানে আমাদের নতুন নোডের এড্রেস দিয়ে দিব। যেহেতু শুরুতেই আমরা আমাদের নতুন নোডের nextNode NULL করে নিয়েছিলাম তাই শেষ নোডের পরবর্তি নোডের এড্রেস NULL-ই থাকবে এখানে।

আর যদি current->nextNode এর মধ্যে থাকা এড্রেস NULL না হয় তার মানে লুপ কোন একটা অবস্থান খুঁজে পেয়ে ব্রেক হয়েছে, লুপের শেষ পর্যন্ত যেতে হয়নি। এখন আমাদেরকে এই নতুন অবস্থানে নোডটিকে সংযোগ দিতে হবে। যেহেতু আমরা লুপের শেষে যাই নি, আমরা একটা অবস্থান পেয়েছি যেই অবস্থানটা হল current এবং current->nextNode এই দুই এড্রেসে থাকা নোড দুইটার মাঝখানে। এদের মাঝে নতুন নোডকে সংযোগ দেওয়ার জন্য যেটা করতে হবে হল নতুন নোডের এড্রেসকে current->nextNode এর মধ্যে রাখতে হবে, আর current->nextNode এ থাকা এড্রেসকে নতুন নোডের nextNode এর মধ্যে রাখতে হবে। সেটিই কোডে করা হয়েছে। এখানে একটা বিষয় খেয়াল করতে হবে, কেন আগে temp->nextNode = current->nextNode করা হয়েছে। আগে current->nextNode = temp করলে কি হত? যেটা হত তা হল আমরা current->nextNode এ থাকা এড্রেসটা হারিয়ে ফেলতাম। ফলে লিংক লিস্টের সংযোগ মাঝখান থেকে ভেঙ্গে যেত। এই জন্য আগে সেই এড্রেসকে temp->nextNode এর মধ্যে রাখা হয়েছে, তারপরে সেখানে temp এ থাকা এড্রেস রাখা হয়েছে। ফলে নতুন নোডটি লিংক লিস্টে তার জায়গা মত সংযুক্ত হয়ে গেল। এরপরে একটা আউটপুট দিয়ে বলা হয়েছে যে কাজটি সম্পন্ন হয়েছে এবং তারপর সম্পূর্ণ লিস্টটি প্রিন্ট করা হয়েছে printList( ) ফাংশন দিয়ে।

লিংক লিস্টে কিভাবে লুপ চালিয়ে এগিয়ে যেতে হয়, কিভাবে সেখানে নতুন নোড সংযোগ করা হয় ইত্যাদি মূল ঝামেলার অংশগুলোই এখানে আলোচনা করা হল। এই ব্যাপারগুলো ব্যবহার করেই ডিলিট এবং প্রিন্ট ফাংশন লিখতে হবে। সেই ফাংশন দুইটা কিভাবে কাজ করে তা বোঝার দায়িত্ব তোমার। সব বলে দিলে নিজের চিন্তা করে শেখাটা হবে না। আর এই ফাংশনটা বুঝলে ঐদুইটা আরও সহজ, নিজে থেকে বোঝার চেষ্টা কর।

ডিলিট কিভাবে করতে হবে তার আইডিয়াটা একটু বলে দেই। যদি যেই ভেলু ডিলিট করতে হবে সেটি head এ থাকে তাহলে তো কেবল পরের নোডটার এড্রেস head এ দিয়ে দিলেই হয়ে যাবে। লিস্টে আগের হেডটা আর থাকল না। আর যদি মাঝ থেকে কোন নোড ডিলিট করতে হয় তাহলে আগের মতই সেই নোড খুঁজে বের করতে হবে। তারপর সেই নোডের মধ্যে যেহেতু পরবর্তি নোডের এড্রেস আছে তাই সেই এড্রেসটাকে আগের নোডের nextNode এর মধ্যে রেখে দিব। ফলে মাঝের নোডটি লিংক লিস্ট থেকে বিচ্ছিন্ন হয়ে গেল। বাকিটুকু নিজে চিন্তা কর।

```c
void deleteNode( int deleteValue )
{
  if( !head ) {
    printf( "No node available yet.\n" );
    return;
  }
```

```
  Node *current, *temp;
  if( deleteValue == head->nodeValue ) {
    temp = head;
    head = head->nextNode;
    free( temp );
  }
  else {
    for( current = head; current->nextNode; current = current->nextNode ) {
      if( current->nextNode->nodeValue == deleteValue )
        break;
    }
    if( !current->nextNode ) {
      printf( "The value was not found in the list.\n" );
      return;
    }
    temp = current->nextNode;
    current->nextNode = current->nextNode->nextNode;
    free( temp );
  }
  printf( "The node has been deleted!\n" );
  printList();
  return;
}
void printList()
{
  Node *current;
  if( !head ) {
    printf( "No node available.\n" );
    return;
  }
  printf( "Here is the updated list:\n" );
  for( current = head; current; current = current->nextNode )
    printf( "%d --> ", current->nodeValue );
  printf( "NULL\n" );
  return;
}
```

লিংক লিস্টের সমগ্র প্রোগ্রামটির কোড কমেন্টসহ পাওয়া যাবে এখানে।