

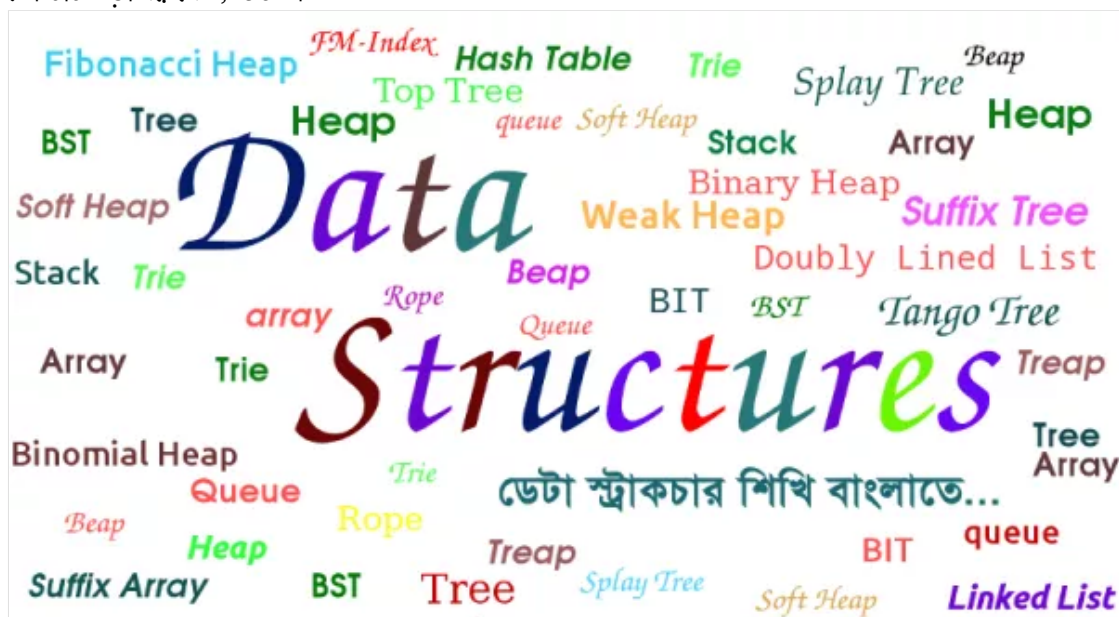


in

G+

হাজারে রাফখাতা

(<https://hellohasan.com/>)



লিংকড লিস্ট – ২ [Singly Linked List Create, insert, delete, search in C]

January 8, 2017

(<https://hellohasan.com/2017/01/08/%e0%a6%b2%e0%a6%bf%e0%a6%82%e0%a6%95%e0%a6%a1->

[%e0%a6%b2%e0%a6%bf%e0%a6%b8%e0%a7%8d%e0%a6%9f-linked-list-create-insert-delete-search/\) / Hasan Abdullah](#)

(<https://hellohasan.com/author/hasan-cse91/>) / Singly Linked List

(<https://hellohasan.com/category/data-structure/linked-list/singly-linked-list/>)

লিংকড লিস্টের প্রথম পর্ব

(<https://hellohasan.com/2017/01/04/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-linked-list-create-print-size/>) থেকে তোমরা এর ব্যাসিক ২-১ টা অপারেশন দেখেছো।

ওখানে ছিল একটা লিংকড লিস্ট তৈরি করে সেটাকে প্রিন্ট করা আর কয়টা আইটেম আছে সেটা count করার অপারেশন। আজ এই পোস্টে আরো কয়েকটা ব্যাসিক অপারেশন নিয়ে আলোচনা করব। সেগুলো হচ্ছে:

- Insert an item at the last position
- Insert an item at the first position
- Insert an item at the middle
- Delete an item
- Search an item

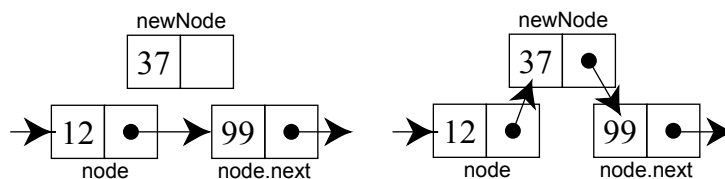
এই পর্বের মূল ফোকাস হচ্ছে insert ও delete. প্রথমেই এই দুটো অপারেশনের আইডিয়াটা ক্লিয়ার করার চেষ্টা করি।

আপনার জেলার ৫ ওয়াক্ত নামাজের শুরু ও শেষের সময় এবং সেহরি ও ইফতারের সময়সূচী জানতে ইন্সটল করুন আমাদের অ্যান্ড্রয়েড অ্যাপ

(<https://play.google.com/store/apps/details?id=theoaktroop.appoframadan>)

Insert an item to a Linked List

লিংকড লিস্ট হচ্ছে একটা নোডের সাথে আরেকটা নোড তাদের নিজেদের মেমরি অ্যাড্রেসের মাধ্যমে যুক্ত থাকা। প্রথম নোডের মধ্যে বলা থাকে দ্বিতীয় নোডের মেমরি অ্যাড্রেস। দ্বিতীয় নোডের মধ্যে বলা থাকে তৃতীয় নোডের memory address. আর শেষ নোডের মধ্যে পরের কোনো নোডের অ্যাড্রেস বলা থাকে না। যেহেতু শেষ নোড, তাই পরের নোডের অ্যাড্রেস হিসেবে বলা থাকে NULL. অর্থাৎ এই নোডের পরে যাওয়ার মত কোন নোড নাই। ধরো শেষের নোডের নাম lastNode. আগের পর্বের মত এই নোডও দুটি ডেটার সমন্বয়ে গঠিত। number ও next. next হচ্ছে পরের নোডের অ্যাড্রেস। যেহেতু এটা শেষ নোড তাই এর অ্যাড্রেসে বলা থাকবে NULL. এখন আমরা যদি এই নোডের পরে আরেকটা নোড যুক্ত করতে চাই তাহলে সিস্টেমটা হবে নতুন একটা নোড তৈরি করা।



Insert a new node into Linked List. Photo from
Wikipedia

ধরো নতুন নোডের নাম দিলাম newNode. এর number variable এ ডেটা অ্যাসাইন করলাম। এরপর এর next ভেরিয়েবলে রাখব NULL. কারণ এই নতুন নোডটাই হতে যাচ্ছে আমাদের লিস্টের শেষ নোড। এই নোডটা কেবল তৈরি হল। এখনো কিন্তু লিস্টে যোগ হয় নি। কার সাথে এটাকে জুড়ে দিতে হবে? আমাদের existing list এর last node এর সাথে এটাকে লিংক করতে পারলেই কিন্তু এটা লিস্টের সাথে যুক্ত হয়ে যাবে। তাই এটাকে লেজ হিসেবে যুক্ত করার জন্য lastNode এর next

এর মান হিসেব অ্যাসাইন করে দিব `newNode` এর মেমরি অ্যাড্রেস। এই `next` এর মান আগে ছিল `NULL`. কিন্তু এখন সে পয়েন্ট করছে `newNode`-কে। ব্যস! আমাদের লিস্টে নতুন একটা নোড যুক্ত হয়ে গেল। মজা না?

একই ঘটনা ঘটবে যদি মাঝের কোন নোডের পরে আমাদের নতুন নোডকে যোগ করতে চাই। উপরের ছবিটা দেখলে আরো পরিষ্কার হবে ব্যাপারটা। প্রথম ছবিতে, প্রথম নোডটা দ্বিতীয় নোডকে পয়েন্ট করে আছে। নতুন একটা নোড বানানো হয়েছে। কিন্তু সেটাকে কেউ পয়েন্ট করে নাই। দ্বিতীয় ছবিতে দেখা যাচ্ছে প্রথম নোডের `next variable` এ রাখা হয়েছে `newNode` এর মেমরি অ্যাড্রেস আর `newNode` এর `next` এ রাখা হয়েছে পরের নোডের `memory address`.

বিসিএস, GRE, ব্যাংক জব, শিক্ষক নিবন্ধন সহ যে কোন চাকুরির পরীক্ষার প্রস্তুতির জন্য ডাউনলোড করুন Editorial Word অ্যাপ্লিকেড অ্যাপ
(<https://play.google.com/store/apps/details?id=megaminds.dailyeditorialword>)

Source Code

`main function` এর উপরে একটা স্ট্রাকচার ডিক্লেয়ার করা হলো এবং এর ভেরিয়েবল ডিক্লেয়ার করা হলো গত পর্বের মত করেই।

Define structure for Linked List

```
1 struct linked_list
2 {
3     int number;
4     struct linked_list *next;
5 };
6
7 typedef struct linked_list node;
8 node *head=NULL, *last=NULL;
```

Insert an item at the Last position of Linked List

কোনো লিস্টের শেষে নতুন কোন আইটেম যোগ করার জন্য নিচের ফাংশনটি ব্যবহার করা যায়। এতে প্রথমে নতুন একটা নোড (`temp_node`) তৈরি করে তাতে ডেটা স্টোর করা হয়েছে। আর `next variable` এ রাখা হয়েছে `NULL` (যেহেতু এটাই শেষ নোড হতে যাচ্ছে)। আর এই নতুন নোডের মেমরি অ্যাড্রেস রাখা হচ্ছে আগের লাস্ট নোডের `next` নামক `variable` এ।

Insert an item at the Last position of Linked List in C

```

1 void insert_at_last(int value)
2 {
3     node *temp_node;
4     temp_node = (node *) malloc(sizeof(node));
5
6     temp_node->number=value;
7     temp_node->next=NULL;
8
9     //For the 1st element
10    if(head==NULL)
11    {
12        head=temp_node;
13        last=temp_node;
14    }
15    else
16    {
17        last->next=temp_node;
18        last=temp_node;
19    }
20
21 }

```

IF condition এ চেক করা হচ্ছে ইনসার্ট করতে চাওয়া নোডটা কি লিস্টের প্রথম নোড কিনা। লিস্টে কোনো আইটেম নাই এমন অবস্থায় যদি এই ফাংশন কল করা হয় তখন এই কন্ডিশনটি কাজ করবে। head এর জন্য memory allocate করা না থাকলে head==NULL এই শর্তটি সত্য হবে। তখন head = temp_node; করার মাধ্যমে head নামক নোডের মেমরি অ্যাড্রেস হিসেবে বলে দেয়া হলো যে, temp_node এর জন্য যেই memory allocate করা হয়েছে সেটাই হবে head এর মেমরি অ্যাড্রেস। আর এটিই যেহেতু প্রথম নোড আর এটাই এখন পর্যন্ত একমাত্র নোড তাই last নোডও head নোডকেই পয়েন্ট করবে।

যদি লিস্টে আগে থেকে এক বা একাধিক নোড থেকে থাকে তাহলে ELSE block-টা কাজ করবে। সেক্ষেত্রে লিস্টের last নোডটার next variable-টা স্টোর করবে temp_node এর মেমরি অ্যাড্রেস। তখন temp_node হয়ে যাবে শেষ নোড, তাই last = temp_node; করার মাধ্যমে last নোডের নিজের মেমরি অ্যাড্রেসকে আপডেট করে দেয়া হল। এই মুহুর্তে last নোডের মেমরি অ্যাড্রেসের মাধ্যমে number প্রিন্ট করতে চাইলে দেখা যাবে temp_node এর number-ই প্রিন্ট করবে। কারণ দুইটা নোডের আলাদা নাম হলেও এরা মূলত একই মেমরি অ্যাড্রেসের একটা নোডকে পয়েন্ট করে আছে।

Insert an item at the first position of Linked List

কোনো একটা লিস্টের শুরুতে যদি একটা নোড যোগ করতে চাই সেক্ষেত্রে নতুন একটা নোড বানাতে হবে। সেই নোডের next এ assign করতে হবে head এর মেমরি অ্যাড্রেস। এরপর head এর অ্যাড্রেসও চেঞ্জ করে দিতে হবে নতুন নোডের অ্যাড্রেস দিয়ে।

Insert an item at the first position of Linked List in C

```

1 void insert_at_first(int value)
2 {
3     node *temp_node = (node *) malloc(sizeof(node));
4
5     temp_node->number=value;
6     temp_node->next = head;
7
8     head = temp_node;
9 }

```

Insert an item middle in the Linked List

আমাদের উদ্দেশ্য হচ্ছে লিস্টের কোনো একটা value (number) এর পরে নতুন একটা নোড insert করা।

ধরো লিস্ট A নামের একটা নোড আছে। এই নোডের number = key. next ভেরিয়েবলটা স্টোর করছে B নামের আরেকটা নোডের মেমরি অ্যাড্রেস। অর্থাৎ A->next = B. আমরা চাই যেই নোডের number হিসেবে key রয়েছে এরপরে নতুন একটা নোড যোগ করতে।

তাহলে A->next = newNode (অর্থাৎ A নোডটা পয়েন্ট করবে newNode-কে), newNode->next = B (অর্থাৎ newNode টা পয়েন্ট করছে B নোডকে)। এভাবে আমরা একটা নতুন নোডকে লিস্টের মাঝে যোগ করে দিতে পারি।

```
Insert an item middle in the Linked List in C
1 void insert_after(int key, int value)
2 {
3     node *myNode = head;
4     int flag = 0;
5
6     while(myNode!=NULL)
7     {
8         if(myNode->number==key)
9         {
10            node *newNode = (node *) malloc(sizeof(node));
11            newNode->number = value;
12            newNode->next = myNode->next;
13            myNode->next = newNode;
14
15            printf("%d is inserted after %d\n", value, key);
16
17            flag = 1;
18
19            break;
20        }
21        else
22            myNode = myNode->next;
23    }
24
25    if(flag==0)
26        printf("Key not found!\n");
27
28 }
```

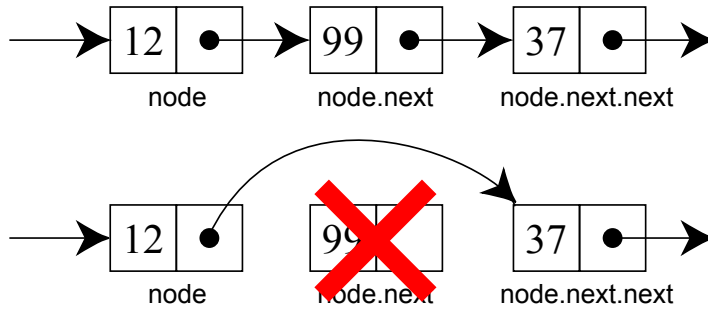
উপরের ফাংশনে, myNode পয়েন্ট করেছে head নোডকে। উদ্দেশ্য হচ্ছে head-এর মাধ্যমে পুরো লিস্টে key খুঁজে দেখা। যেই নোডের number হিসেবে key-কে পাওয়া যাবে সেই নোডের পরেই নতুন নোড যুক্ত হবে। সেই কাজটাই করা হয়েছে ৮ নাম্বার লাইনের IF block এ। এই ব্লকে নতুন একটা নোডের (newNode) জন্য memory allocate করা হয়েছে। newNode এর number এ value বসানো হয়েছে। আর next এ বসানো হয়েছে myNode এর next এর ভ্যালু। অর্থাৎ myNode যেই নোডটাকে আগে পয়েন্ট করত সেটাকে এখন থেকে পয়েন্ট করবে newNode. আর myNode এখন থেকে পয়েন্ট করবে newNodeকে। তাই myNode এর next এর মানও আপডেট করা হয়েছে newNode এর মেমরি অ্যাড্রেস দিয়ে।

IF যদি সত্য না হয় তাহলে ELSE এর myNode = myNode->next; এর মাধ্যমে লিস্টের পরের নোডের মেমরি অ্যাড্রেসকে myNode পয়েন্ট করবে। এরপর লুপ ঘুরে আবার চেক করবে পরের নোডের number==key কিনা? যদি কখনো number==key পাওয়া যায় তাহলে উপরের বর্ণনা অনুযায়ী IF block কাজ করবে। কাজের শেষে flag = 1 করে দিয়ে লুপ break করবে। যদি number==key পাওয়া না যায় তাহলে flag এর মান 0-ই থেকে যাবে। লুপের বাইরে এসে আরেকটা চেক করবে। flag == 0 হলে অর্থাৎ key খুঁজে না পেলে প্রিন্ট করে দিবে “Key not found”.

অ্যারের মাঝে যদি কোন আইটেম যোগ করতে চাই তাহলে প্রসেসটা কিন্তু লিংকড লিস্টের চেয়ে জটিল আর slow। কারণ অ্যারের মাঝে কোনো একটা নতুন আইটেম যোগ করতে চাইলে যেই ইনডেক্সে যোগ করতে চাই তার পরের সবগুলো ইনডেক্সের ভ্যালুগুলোকে এক ঘর করে পেছনে সরাতে হবে। কিন্তু লিংকড লিস্টের ক্ষেত্রে সেই ঝামেলা করা লাগছে না। তাই অনেকটা runtime বেঁচে যাচ্ছে।

Delete an item from Linked List

ধরো A, B, C তিনটা নোড। A পয়েন্ট করে আছে B কে, B পয়েন্ট করে আছে C কে। আমরা B কে লিস্ট থেকে ডিলেট করতে চাই। তাহলে সহজ কাজটা হলো A কে পয়েন্ট করতে দিব C কে। A যদি C কে পয়েন্ট করে তাহলে B কে কেউ পয়েন্ট করছে না। এই বেচারি এমনভাবেই লিস্টের বাইরে চলে যাবে।



Delete a node from Linked List. Photo from Wikipedia

Source Code

ফাংশনে প্যারামিটার হিসাবে value পাঠানো হচ্ছে। প্রথম যেই নোডের number এর মান value এর সমান হবে সেই নোডটাকে ডিলেট করা হবে। এজন্য value এর সাথে match করা নোডের অ্যাড্রেস, এই নোডের আগের নোডের অ্যাড্রেস ও পরের নোডের অ্যাড্রেস জানা থাকা লাগবে। কারণ আমাদের লক্ষ্য হচ্ছে সংশ্লিষ্ট নোডের পরের নোডের সাথে এর আগের নোডের লিংক করায় দেয়া। যেন মাঝ থেকে নোডটা ডিলেট হয়ে যায়।

Delete an item from Linked List in C

```

1 void delete_item(int value)
2 {
3     node *myNode = head, *previous=NULL;
4     int flag = 0;
5
6     while(myNode!=NULL)
7     {
8         if(myNode->number==value)
9         {
10             if(previous==NULL)
11                 head = myNode->next;
12             else
13                 previous->next = myNode->next;
14
15             printf("%d is deleted from list\n", value);
16
17             flag = 1;
18             break;
19         }
20
21         previous = myNode;
22         myNode = myNode->next;
23     }
24
25     if(flag==0)
26         printf("Key not found!\n");
27 }

```

myNode পয়েন্ট করে আছে head-কে। myNode এর মাধ্যমে পুরো লিস্ট traverse করে দেখা হবে myNode->number==value পাওয়া যায় কিনা। যদি প্রথম নোডেই পাওয়া না যায় তাহলে previous = myNode; এবং myNode = myNode->next; করা হল। কারণ প্রথম নোডে পাওয়া না গেলে পরের নোডে খুঁজতে হবে। পরের নোডে যাওয়ার ফলে, myNode-টা কিন্তু previous node হয়ে যাবে। তাই previous = myNode করা হল। আর পরের নোডে যাওয়ার জন্য myNode = myNode->next করা হলো।

যদি value খুঁজে পাওয়া যায় তাহলে প্রথমেই চেক করা হচ্ছে previous==NULL কিনা। এটা সত্য হবার মানে হচ্ছে লিস্টের প্রথম নোডেই value পাওয়া গেছে। head এ যদি ভ্যালু পাওয়া যায় এর আগে কিন্তু কোনো নোড নাই। তাই previous এর মান NULL. সেক্ষেত্রে head=myNode->next. অর্থাৎ head শুরুতে যেই নোডকে পয়েন্ট করত (দ্বিতীয় নোড), সেই নোডটাই এখন হয়ে গেল head node. দ্বিতীয় নোডটাই head হয়ে যাওয়াতে আগের head-টা ভ্যানিস হয়ে যাবে।

আর value-টি প্রথম নোডেই পাওয়া না গেলে ১৩ নম্বরের লাইনটা কাজ করবে previous->next = myNode->next. অর্থাৎ value টা পাওয়া গেছে myNode এ। একে ডিলেট করতে previous নোডকে পয়েন্ট করতে বলা হচ্ছে myNode এর পরের নোডকে (myNode->next). ফলে previous node টি পয়েন্ট করছে myNode এর পরের নোডকে। myNode-কে কেউ পয়েন্ট করছে না, তাই এটি ডিলেট হয়ে যাবে।

এই ডিলেট অপারেশনটা অ্যারের কোনো আইটেম ডিলেটের চেয়ে efficient. কারণ অ্যারের মাঝ থেকে কোনো একটা আইটেম ডিলেট করতে হলে সেই আইটেমের পরের সকল আইটেমকে এক ঘর করে বাম দিকে সরিয়ে নিয়ে আসতে হয়। কিন্তু লিংকড লিস্টে সব আইটেমকে সরানো দরকার হচ্ছে না। দুইটা নোডের লিংক চেঞ্জ করে দিলেই খেল খতম!

Search an item from Linked List

উপরের insert আর delete বুঝে থাকলে search বুঝতে সমস্যা হবে না। পরোক্ষ ভাবে কিন্তু আমরা ইনসার্ট, ডিলেট উভয় ক্ষেত্রেই সার্চের কাজটা করেছি।

Search an item from Linked List in C

```

1 void search_item(int value)
2 {
3     node *searchNode = head;
4     int flag = 0;
5
6     while(searchNode!=NULL)
7     {
8         if(searchNode->number==value)
9         {
10             printf("%d is present in this list. Memory address is %d\n", value, searchNode);
11             flag = 1;
12             break;
13         }
14         else
15             searchNode = searchNode->next;
16     }
17
18     if(flag==0)
19         printf("Item not found\n");
20
21 }

```

ইনসার্ট-ডিলেটের মত করে head থেকে লুপ ঘুরিয়ে সার্চ করা হচ্ছে। কোনো নোডের মেমরি অ্যাড্রেস হিসেবে NULL পাওয়ার আগ পর্যন্ত এই সার্চিং চলতে থাকবে। যদি value-কে খুঁজে পাওয়া যায় তাহলে একটা মেসেজ প্রিন্ট করে loop break করে দেয়া হচ্ছে। খুঁজে না পেলে লুপের বাইরে এসে প্রিন্ট করা হচ্ছে “Item not found”.

পুরো কোডটা পাওয়া যাবে আমার **গিটহাব রিপোজিট**রিতে

([https://github.com/hasancse91/data-](https://github.com/hasancse91/data-structures/blob/master/Source%20Code/Linked%20List%20%5Bcreate%2C%20insert%2C%20delete%2C%20search%5D.c)


[structures/blob/master/Source%20Code/Linked%20List%20%5Bcreate%2C%20insert%2C%20delete%2C%20search%5D.c](https://github.com/hasancse91/data-structures/blob/master/Source%20Code/Linked%20List%20%5Bcreate%2C%20insert%2C%20delete%2C%20search%5D.c))।

কোথাও কোনো ভুল চোখে পড়লে বা মতামত থাকলে কমেন্ট করতে ভুলবে না। এত বড় পোস্টটা পড়ার জন্য ধন্যবাদ। আদৌ কিছু বুঝাতে পেরেছি? :’(


Share this:

 Facebook


(<https://hellohasan.com/2017/01/08/%e0%a6%b2%e0%a6%bf%e0%a6%82%e0%a6%95%e0%a6%a1-%e0%a6%b2%e0%a6%bf%e0%a6%b8%e0%a7%8d%e0%a6%9f-linked-list-create-insert-delete-search/?share=facebook&nb=1>)

 WhatsApp (whatsapp://send?)

text=%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1%20%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F%20-%20%E0%A7%A8%20%5BSingly%20Linked%20List%20Create%2C%20insert%2C%20delete%2C%20search%20in%20C%5D
<https://3A%2F%2Fhellohasan.com%2F2017%2F01%2F08%2F%25e0%25a6%25b2%25e0%25a6%25bf%25e0%25a6%2582%25e0%25a6%2595%25e0%25a6%25a1-%25e0%25a6%25b2%25e0%25a6%25bf%25e0%25a6%25b8%25e0%25a7%258d%25e0%25a6%259f-linked-list-create-insert-delete-search%2F>)

 Skype (<https://hellohasan.com/2017/01/08/%e0%a6%b2%e0%a6%bf%e0%a6%82%e0%a6%95%e0%a6%a1-%e0%a6%b2%e0%a6%bf%e0%a6%b8%e0%a7%8d%e0%a6%9f-linked-list-create-insert-delete-search/?share=skype&nb=1>)

share=skype&nb=1)

 Google (<https://hellohasan.com/2017/01/08/%e0%a6%b2%e0%a6%bf%e0%a6%82%e0%a6%95%e0%a6%a1-%e0%a6%b2%e0%a6%bf%e0%a6%b8%e0%a7%8d%e0%a6%9f-linked-list-create-insert-delete-search/?share=google-plus-1&nb=1>)

share=google-plus-1&nb=1)

 LinkedIn 5

(<https://hellohasan.com/2017/01/08/%e0%a6%b2%e0%a6%bf%e0%a6%82%e0%a6%95%e0%a6%a1-%e0%a6%b2%e0%a6%bf%e0%a6%b8%e0%a7%8d%e0%a6%9f-linked-list-create-insert-delete-search/?share=linkedin&nb=1>)

 Pocket (<https://hellohasan.com/2017/01/08/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-INSERT-DELETE-SEARCH/?share=pocket&nb=1>)

 Reddit (<https://hellohasan.com/2017/01/08/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-INSERT-DELETE-SEARCH/?share=reddit&nb=1>)

 Twitter (<https://hellohasan.com/2017/01/08/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-INSERT-DELETE-SEARCH/?share=twitter&nb=1>)

 Telegram (<https://hellohasan.com/2017/01/08/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-INSERT-DELETE-SEARCH/?share=telegram&nb=1>)

 Pinterest (<https://hellohasan.com/2017/01/08/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-INSERT-DELETE-SEARCH/?share=pinterest&nb=1>)

 Print (<https://hellohasan.com/2017/01/08/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-INSERT-DELETE-SEARCH/#print>)

 Email (<https://hellohasan.com/2017/01/08/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-INSERT-DELETE-SEARCH/?share=email&nb=1>)

Related

- লিংকড লিস্ট - ১ [Singly Linked List Create & Print in C] (<https://hellohasan.com/2017/01/04/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-PRINT-SIZE/>)

January 4, 2017

In "Singly Linked List"
- লিংকড লিস্ট – ৪ [Doubly Linked List: Delete item from head, tail and middle] (<https://hellohasan.com/2017/06/12/doubly-linked-list-delete-item-from-head-tail-and-middle/>)

June 12, 2017

In "Doubly Linked List"
- লিংকড লিস্ট – ৩ [Doubly Linked List: Insert, Print Forward and Reverse order] (<https://hellohasan.com/2017/05/13/doubly-linked-list-insert-print/>)

May 13, 2017

In "Doubly Linked List"

Tagged C - সি (<https://hellohasan.com/tag/c/>)

লিংকড লিস্ট – ১ [SINGLY LINKED LIST CREATE & PRINT IN C] ([HTTPS://HELLOHASAN.COM/2017/01/04/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-PRINT-SIZE/](https://HELLOHASAN.COM/2017/01/04/%E0%A6%B2%E0%A6%BF%E0%A6%82%E0%A6%95%E0%A6%A1-%E0%A6%B2%E0%A6%BF%E0%A6%B8%E0%A7%8D%E0%A6%9F-LINKED-LIST-CREATE-PRINT-SIZE/))

ড্রি ডেটা স্ট্রাকচার – ১ [BASIC CONCEPT] ([HTTPS://HELLOHASAN.COM/2017/01/15/%E0%A6%9F%E0%A7%8D%E0%A6%B0%E0%A6%BF%E0%A6%A1%E0%A7%87%E0%A6%9F%E0%A6%BE%E0%A6%B8%E0%A7%8D%E0%A6%9F%E0%A7%8D%E0%A6%B0%E0%A6%BE%E0%A6%95%E0%A6%9A%E0%A6](https://HELLOHASAN.COM/2017/01/15/%E0%A6%9F%E0%A7%8D%E0%A6%B0%E0%A6%BF%E0%A6%A1%E0%A7%87%E0%A6%9F%E0%A6%BE%E0%A6%B8%E0%A7%8D%E0%A6%9F%E0%A7%8D%E0%A6%B0%E0%A6%BE%E0%A6%95%E0%A6%9A%E0%A6))

%BE%E0%A6%B0-%E0%A7%A7-BASIC-
CONCEPT/)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

Hit counter

- 251,449 hits

Post Categories

☐ অ্যান্ড্রয়েড অ্যাপ ডেভেলপমেন্ট

(<https://hellohasan.com/category/android-tutorial/>) (15)

☐ Android Library

(<https://hellohasan.com/category/android-tutorial/popular-android-library/>) (7)

☐ Development Tools

(<https://hellohasan.com/category/android-tutorial/android-development-tools/>) (2)

☐ Essential Topics

(<https://hellohasan.com/category/android-tutorial/essential-android-tutorial/>) (6)

☐ কম্পিউটার সায়েন্স ও আইটি

(<https://hellohasan.com/category/computer-science-it-industry/>) (11)

☐ প্রযুক্তি

(<https://hellohasan.com/category/technology/>) (3)

☐ প্রোগ্রামিং

(<https://hellohasan.com/category/programming/>) (6)

☐ অনলাইন জাজ সিরিজ

(<https://hellohasan.com/category/online-programming-judge-series/>) (13)

☐ ডেটা স্ট্রাকচার-Data Structure

(<https://hellohasan.com/category/data-structure/>) (21)

☐ সাধারণ আলোচনা

(<https://hellohasan.com/category/data-structure/data-structure-introduction/>) (1)

☐ অ্যারে – Array

(<https://hellohasan.com/category/data-structure/array/>) (2)

☐ স্ট্যাক – Stack

(<https://hellohasan.com/category/data-structure/stack/>) (3)

□ কিউ – Queue

(<https://hellohasan.com/category/data-structure/queue/>) (2)

□ লিংকড লিস্ট – Linked List

(<https://hellohasan.com/category/data-structure/linked-list/>)

(6)

□ Singly Linked List

(<https://hellohasan.com/category/data-structure/linked-list/singly-linked-list/>) (2)

□ Doubly Linked List

(<https://hellohasan.com/category/data-structure/linked-list/doubly-linked-list/>) (2)

□ Circular Linked List

(<https://hellohasan.com/category/data-structure/linked-list/circular-linked-list/>) (2)

□ ট্রি – Tree

(<https://hellohasan.com/category/data-structure/tree/>) (7)

□ ব্যাসিক কনসেপ্ট

(<https://hellohasan.com/category/data-structure/tree/tree-basic-concept/>) (2)

□ বাইনারি সার্চ ট্রি – BST

(<https://hellohasan.com/category/data-structure/tree/binary-search-tree-bst/>) (5)

□ অ্যালগরিদম – Algorithm

(<https://hellohasan.com/category/algorithm/>) (9)

□ সাধারণ আলোচনা

(<https://hellohasan.com/category/algorithm/algorithm-introduction/>) (2)

□ সার্চিং অ্যালগরিদম

(<https://hellohasan.com/category/>)

ry/algorithm/searching-
algorithm/) (2)

☐ সর্টিং অ্যালগরিদম

(https://hellohasan.com/category/algorithm/sorting-
algorithm/) (4)

☐ গ্রাফ অ্যালগরিদম

(https://hellohasan.com/category/algorithm/graph-algorithm/)
(1)

☐ সংখ্যা পদ্ধতি সিরিজ

(https://hellohasan.com/category/number-system-conversion-series/)
(11)

☐ অনুপ্রেরণা

(https://hellohasan.com/category/inspiration/) (3)

☐ জন সচেতনতা

(https://hellohasan.com/category/public-awareness/) (6)

Facebook



Me on Facebook

(https://www.facebook.com/hasan.cse91)

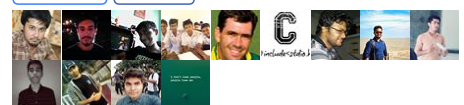
Facebook Page of my Blog

(https://www.facebook.com/HasanerRafkhata/)

Like

Share

You and 9.8K others like this.



সাম্প্রতিক জনপ্রিয় পোস্টসমূহ



(<https://hellohasan.com/2017/01/15/%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf-%e0%a6%a1%e0%a7%87%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0-%e0%a7%a7-basic-concept/>)

ড্রি ডেটা স্ট্রাকচার - ১ [Basic Concept]

(<https://hellohasan.com/2017/01/15/%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf-%e0%a6%a1%e0%a7%87%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0-%e0%a7%a7-basic-concept/>)



(<https://hellohasan.com/2016/09/05/computer-science-job-opportunities-in-bangladesh/>)

প্রোগ্রামিং ছাড়াও CSE
গ্র্যাজুয়েটদের আছে অনেক
চাকুরি

(<https://hellohasan.com/2016/09/05/computer-science-job-opportunities-in-bangladesh/>)



(<https://hellohasan.com/2017/06/03/software-engineer-preparation-and-experience/>)

সফটওয়্যার ইঞ্জিনিয়ার হবার
জন্য আমার প্রস্তুতি ও গত দেড়
মাস চাকুরির অভিজ্ঞতা
(<https://hellohasan.com/2017/06/03/software-engineer-preparation-and-experience/>)

(<https://hellohasan.com/2017/01/15/%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0-applications-classification/>)



(<https://hellohasan.com/2017/01/15/%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0-applications-classification/>)

ড্রি ডেটা স্ট্রাকচার - ২

[Applications and
Classification]

(<https://hellohasan.com/2017/01/15/%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0-applications-classification/>)



(<https://hellohasan.com/2017/10/01/android-retrofit-get-post-method-different-network-layer/>)

Android এ Retrofit
ব্যবহার করে GET ও POST
রিকোয়েস্ট [different
network layer] - ২

(<https://hellohasan.com/2017/10/01/android-retrofit-get-post-method-different-network-layer/>)



(<https://hellohasan.com/2017/01/15/%e0%a6%ac%e0%a6%be%e0%a6%87%e0%a6%a8%e0%a6%be%e0%a6%b0%e0%a6%bf-%e0%a6%b8%e0%a6%be%e0%a6%b0%e0%a7%8d%e0%a6%9a-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf-binary-search-tree-bst/>)

ট্রি ডেটা স্ট্রাকচার - ৩

[বাইনারি সার্চ ট্রি - BST]

(<https://hellohasan.com/2017/01/15/%e0%a6%ac%e0%a6%be%e0%a6%87%e0%a6%a8%e0%a6%be%e0%a6%b0%e0%a6%bf-%e0%a6%b8%e0%a6%be%e0%a6%b0%e0%a7%8d%e0%a6%9a-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf-binary-search-tree-bst/>)



(<https://hellohasan.com/2017/07/16/android-app-development-guideline/>)

Android App ডেভেলপমেন্ট
গাইড লাইন

(<https://hellohasan.com/2017/07/16/android-app-development-guideline/>)



(<https://hellohasan.com/2016/10/07/%e0%a6%a1%e0%a7%87%e0%a6%9f%e0%a6%be->

%e0%a6%b8%e0%a7%8d%e0%
a6%9f%e0%a7%8d%e0%a6%b0
%e0%a6%be%e0%a6%95%e0%
a6%9a%e0%a6%be%e0%a6%b0
-

%e0%a6%aa%e0%a6%b0%e0%
a6%bf%e0%a6%9a%e0%a7%9f/
)

ডেটা স্ট্রাকচার কী ও কেন?

(https://hellohasan.com
/2016/10/07/%e0%a6%a
1%e0%a7%87%e0%a6
%9f%e0%a6%be-
%e0%a6%b8%e0%a7%
8d%e0%a6%9f%e0%a7
%8d%e0%a6%b0%e0%
a6%be%e0%a6%95%e0
%a6%9a%e0%a6%be%
e0%a6%b0-
%e0%a6%aa%e0%a6%
b0%e0%a6%bf%e0%a6
%9a%e0%a7%9f/)



(https://hellohasan.com/2
016/10/20/%e0%a6%ac%
e0%a6%be%e0%a6%87%
e0%a6%a8%e0%a6%be%e0%a6
%b0%e0%a6%bf-
%e0%a6%b8%e0%a6%be%e0%
a6%b0%e0%a7%8d%e0%a6%9a
-binary-search-algorithm/)

বাইনারি সার্চ অ্যালগরিদম -

Binary Search

Algorithm

(https://hellohasan.com
/2016/10/20/%e0%a6%a
c%e0%a6%be%e0%a6
%87%e0%a6%a8%e0%
a6%be%e0%a6%b0%e0
%a6%bf-
%e0%a6%b8%e0%a6%
be%e0%a6%b0%e0%a7
%8d%e0%a6%9a-
binary-search-
algorithm/)

(<https://hellohasan.com/2016/08/15/8-barriers-to-overcome-when-learning-to-code/>)



প্রোগ্রামিং শেখার সময় জয় করতে হবে ৮ টি প্রতিবন্ধকতা
(<https://hellohasan.com/2016/08/15/8-barriers-to-overcome-when-learning-to-code/>)

Custom Ad: App of Ramadan 2017



(<https://goo.gl/mCYFRh>)

সাহরি-ইফতার ও সারা বছরের নামাজের সময়সূচী জানার জন্য ব্যবহার করুন আমার টিমের ডেভেলপ করা Android App!
(<https://goo.gl/mCYFRh>)

Ad – 1



(<https://goo.gl/hzOJkR>)

Ad – 2

ডোমেইন হোস্টিং সার্ভিসের সেরা প্যাকেজগুলো
দিচ্ছে Techno Haat!

(<https://goo.gl/mTGZLe>)



(<https://goo.gl/mTGZLe>)



Proudly powered by WordPress (<https://wordpress.org/>) | Theme: Amadeus (<http://themeisle.com/themes/amadeus/>) by Themeisle.