به نام خدا

توضیحات پروژه و روند اجرای برنامه

امیرمحمد پیرحسین لو

۹۵۳۱۰۱۴

نکته ی اول :

در این برنامه در صورت اضافه شدن subservice اگر زیر خدمتی با همین نام قبلا در Glist وجود داشته باشد ، خود برنامه در صورت اضافه کردن آن به GList ، یک Index هم به آن نسبت می دهد تا بین این زیرخدمت و زیرخدمت قبلی تمایز قائل شویم . مثلا اگر ۵ زیرخدمت با نام a داشته باشیم ، آن ها به صورت زیر در Glist ذخیره می شوند :

a with index 0
a with index 1
a with index 2
a with index 3
a with index 4

بدیهی است که هنگام سفارش دادن یک خدمت یا زیر خدمت توسط کاربر نیاز است که کاربر Index خدمت یا زیر خدمتی را که میخواهد وارد کند ، پس در این صورت در این برنامه برای سفارش دادن باید فرمت دستور به صورت زیر باشد :

order <Service_Name> **to** <Agency_Name> **by** <Customer_Name> **with** <Immediacy_Level> **with** <Index>

add subservice <Subservice_Name> **to** <Service_Name> **with** <Index>

همیچنین نیازی نیست تا کاربر Index ها را حفظ کند . کاربر می تواند با زدن دستور زیر تمام خدمات و زیرخدمات را همراه با Index هر یک مشاهده کند .

all

همچنین کاربر می تواند با استفاده از دستور زیر سرویس هایی که توسط نمایندگی ها ارائه می شوند را مشاهده کند .

list services

توجه : تمام سرویس های **اصلی** دارای index برابر صفر هستند و اضافه کردن سرویس اصلی که در حال حاضر در GList قرار دارد **مجاز نیست** .

توضیحات توابع + **time complexity**

1-Service.class:

a)

```
/**
 * Create a Node which can be interpreted as Service or Sub Service.
 * This method is used while adding a Service or Sub Service .
 * @param service_name  name of Service or Sub Service
 * @param i             is used for distinguish between 2 Nodes which have
same names
 * @return              the address of new Node
 */
Node *createNode(string service_name, int i);
```

به وضوح مرتبه زمانی این تابع O(1) است .

b)

```
/**
 * Find parent of Sub Service.
 * More information in PDF file.
 * @param service_name
 * @param sub_service_name
 * @param start
 * @return
 */
Node *findParent(string service_name, string sub_service_name, Node *start,
int ind);
```

در این تابع به دنبال پدر زیرخدمتی که قرار است اضافه شود می گردیم ، با پیمایش ، یک مسیر ***یکتا*** برای رسیدن به پدر به دست می آید . اگر دراین مسیر خود زیر خدمت مشاهده شد ، -1 برگردانده می شود و اگر پدر پیدا نشد ، 0 برگردانده می شود . در صورت پیدا شدن پدر ، آدرس آن برگردانده می شود .

مرتبه ی زمانی این الگوریتم در بدترین حالت ، O (n + e) است که n تعداد کل node ها در GList اعم از خدمت و زیرخدمت است و e تعداد node ها در کوتاه ترین مسیر بین root و پدر زیرخدمت است . از آنجا که n>=e ، پس مرتبه زمانی O(n) است . این مرتبه زمانی به دلیل صدا زدن تابع findParent دیگری که در زیر می آید به دست می آید.

c)

```
/**
 * Find Parent of Sub Service with using an algorithm like DFS algorithm in
Graph theory.
 * also set the path from root to parent in stack for other operations.
 * @param name
 * @param p
 * @param s
 * @param ind
 * @return
 */
Node *findParent(string name, Node *p, stack<Node *> *s, int ind);
```

مرتبه زمانی تابع O(n) است به همان دلایل که در قسمت قبل گفته شد .

d)

```
/**
 * Print information of all Sub Services of a Node.
 * In other words, print information of all children of a Node and
 * recursively print information of children of that children.
 * @param p     starting point of print
 */
void print(Node *p);
```

از آن جا که هر node نهایتا n – 1 بچه دارد ، پس مرتبه زمانی تابع برابر است با O(n)

e)

```
/**
 * Delete all Sub Services of a Node and itself
 * @param p     starting point of delete
 */
void deleteNode(Node *p);
```

تنها فرق این تابع با تابع print که در بالا به آن اشاره شد این است که در print ، برای هر node اطلاعات چاپ می شود ولی در این تابع هر node پاک می شود . پس مرتبه زمانی آن مشابه مرتبه زمانی تابع print یعنی O(n) است .

f)

```
/**
 * Add a Service to Generalized Linked List by creating a new Node.
 * with handling Errors which may occur during adding Service to
Generalized Linked List.
 * Time Complexity: O(1) with ignoring number of Services in GList.
 * @param service_name      name of service for passing to createNode
function .
 */
void addService(string service_name);
```

g)

```
/**
* Add a Sub Service to a Service with handling Errors which may occur
* during the operation .
* For example , prevent creation of loop by adding this Sub Service to
intended Service.
* This Handling is implemented by using stack.
* also
* if there is a Sub Service in Generalized Linked List with same name to
new Sub Service,
* we distinguish between these Sub Services by adding identifier to those
Nodes.
* Time Complexity: in worst case , we most search all Nodes in GList to
find parent,
* so O(n) , n number of Nodes in GList.
* More information in PDF file.
* @param sub_service_name
* @param parent
* @param ind
*/
void addSubService(string sub_service_name, string parent, int ind);
```

h)

```
/**
 * Return a pointer to mentioned Service with name service_name
 * if there is no Service named service_name , return NULL.
 * The caller of this method is an agency ,
 * so service->agency++;
 * Time complexity: O(n)
 * n : number of Services in GList.
 * @param service_name
 * @return Node*
 */
Node *getRefrenceToMainService(string service_name);
```

i)

```
/**
 * Find a service and delete it and its children by calling method
deleteNode.
 * Time complexity: is same as time complexity of listServicesFrom method ,
 * that's mean O ( k + n )
 * k : number of Services in GList.
 * n : number of all Nodes in GList.
 * reason same as the reasons in documentation of listServicesFrom method
 * @param service_name
 */
void deleteMainService(string service_name);
```

j)

```
/**
 * Print all sub Services of the intended Service by calling print method.
 * time complexity: O(k + n )
 * k is Number of Services . O(k) for searching intended Service.
 * O(n) is time complexity of print method . n is Number of Sub Services of
intended Service.
 * Because of our GList has long depth and short width , n is approximately
equal with number of all Nodes in GList.
 * For checking time complexity , this assumption is true.
 * @param service_name
 */
void listServicesFrom(string service_name);
```

k)

```
/**
 * Checks Service or Sub Service with service_name exists in Generalized
Linked List or not
 * This operation is implemented by using this method recursively.
 * if exists , returning true
 * else returning false
 * For doing this operation in worst case , all Nodes in Generalized list
most be visited
 * so the time complexity is O(n) . n is number of all Nodes in generalized
Linked List.
 * @param service_name
 * @param parent
 * @return boolean
 */
bool contain(string service_name, Node *parent, int identifier);
```

l)

```
/**
 * Print all services of Generalized Linked List.
 * This process is executed by calling DFSServices method on each service.
 */
void listServices();
```

مرتبه زمانی این تابع برابر $O(n)$ است ، چون باید کل لیست را چاپ کنیم .

m)

```
/**
 * Iterate on all Services and print Sub Services names.
 * Algorithm which used for doing this operation is like DFS algorithm for
calibration on a graph.
 * @param p
 */
void DFSServices(Node *p);
```

دقیقا مانند مورد print در بخش d) کار می کند . مرتبه زمانی آن در بدترین حالت برابر $O(n)$ است . می توان از تابع print به جای آن استفاده کرد .

2-Agency.class

a)

```
/**
 * Find intended agency and return a reference to it.
 * @param agency_name
 * @return
 */
ANode *whereIsAgency(string agency_name);
```

مرتبه ی زمانی این الگوریتم $O(k)$ است که k تعداد agency ها است .

b)

```
/**
 * Is there any Service with name service_name among services of intended
agency?
 * if yes , return true
 * else return false.
 * also determine location of Service among services Vector of intended
agency.
 * @param service_name
 * @param agency
 * @param loc
 * @return
 */
bool contain(string service_name, ANode *agency, int &loc);
```

مرتبه زمانی در بدترین حالت برابر $O(k)$ است که k برابر سرویس های اصلی است که agency مورد نظر ارائه می دهد.

c)

```
/**
 * Add an agency to linked list of agencies.
 * also prevent form adding two Agency with same name.
 * @param agency_name
 */
void addAgency(string agency_name);
```

مرتبه زمانی در بدترین حالت $O(n)$ است که n تعداد agency های موجود در لیست پیوندی agency ها است .

d)

```
/**
 * Print all agencies names.
 */
void listAgencies();
```

مرتبه زمانی مانند تابع قسمت c برابر $O(n)$ است که n تعداد agency ها در لیست پیوندی است .

e)

```
/**
 * Print all services of each agency respectively.
 * This process is executed by calling DFSServices method on each service
of each agency.
 */
void listServices();
```

در صورتی ک هر agency در کل n خدمت (خدمت یا زیر خدمت) ارائه دهد و $\log n$ تا سرویس اصلی ارائه دهد ،
مرتبه زمانی $O(knlogn)$ است که k تعداد agency های لیست پیوندی است . در غیر این صورت در بد ترین حالت

O(n*k*g) است که g تعداد سرویس های اصلی و n به طور میانگین تعداد بچه های این سرویس های اصلی است .

f)

```
/**
 * Add an service to service Vector of intended agency.
 * with handling undefined behaviours like adding a service twice to an
agency , ...
 * @param service_name
 * @param agency_name
 */
void addOffer(string service_name, string agency_name);
```

در بدترین حالت مرتبه زمانی اجرای این تابع برابر با جمع مرتبه زمانی اجرای توابع whereIsAgency و contain و getRefrenceToMainService است که برابر O(n) است که n برابر ماکسیمم تعداد agency ها و تعداد service ها است .

g)

```
/**
 * Delete a service from service Vector of agency
 * also if no agency points to that service , delete service
 * by calling deleteMainService Method of Service Class.
 * @param service_name
 * @param agency_name
 */
void deleteService(string service_name, string agency_name);
```

مرتبه زمانی اجرای این تابع برابر با جمع مرتبه زمانی اجرای توابع whereIsAgency و contain و deleteMainService است که برابر می شود با O(n) .

h)

```
/**
 * Order to agency.
 * Create an instance of Order and set values and add it to order Vector of
agency.
 * with handling errors which may occur.
 * After operation , heapify the order vector.
 * @param service_name
 * @param agency_name
 * @param customer_name
 * @param priority
 * @param time
 */
void order(string service_name, string agency_name, string customer_name,
int priority,
           __int64 time, int identifier);
```

اگر بخواهیم چک کردن برای حالات خاصی که مطلوب نیست را کنار گذاریم ، مرتبه ی زمانی اجرای تابع برابر O(1) است .
درغیر این صورت در بدترین حالت O(n) است که n برابر تعداد کل خدماتی است که توسط شرکت ارائه می شود .

i)

```
/**
 * Printing Orders of agency.
 * Time Complexity is O( n log n ) . because at first iterate on Vector of
orders - O(n)
 * then heapify the max_heap of agency - O(nlogn).
 * @param agency_name
 */
void listOrder(string agency_name);
```

مرتبه ی زمانی تابع برابر O (nlogn + n) است برابر O(nlogn) می شود . این مرتبه زمانی نیز به خاطر heapify است .
الگوریتم heapify در کد همراه doc قرار دارد.

---

commands and Test case

add service <Service_Name>

example: **add service** a

add subservice <Subservice_Name> **to** <Service_Name> **with** <Index>

example: **add subservice** b **to** a **with** 0

add agency <Agency_Name>

example: **add agency** 1

add offer <Service_Name> **to** <Agency_Name>

example: **add offer** a **to** 1

list agencies

example : **list agencies**

list services

example : **list services**

list services from <Service_Name>

example : **list services from** a

list orders <Agency_Name>

example : **list orders** 1

delete <Service_Name> **from** <Agency_Name>

example : **delete** a **from** 1

order <Service_Name> **to** <Agency_Name> **by** <Customer_Name> **with** <Immediacy_Level> **with** <Index>

example : **order** a **to** 1 **by** amirphl **with** 1 **with** 0

Sample : (Test case)

```cpp
#include <iostream>
#include <ctime>
#include <chrono>
#include "Service.h"
#include "Agency.h"

class milliseconds;

using namespace std;

int main() {
    Service *service = new Service();
    Agency *agency = new Agency(service);

    service->addService("A");
    service->addService("B");
    service->addService("C");
    service->addService("D");
    service->addSubService("E", "A", 0);
    service->addSubService("F", "A", 1);
    service->addSubService("F", "A", 0);
    service->addSubService("Z", "A", 1);
    service->addSubService("A", "E", 0);
    service->addSubService("B", "F", 0);
    service->addSubService("K", "F", 0);
    service->addSubService("B", "F", 0);
    service->addSubService("G", "K", 0);
    service->addSubService("G", "K", 1);
    service->addSubService("A", "G", 0);
    service->addSubService("G", "C", 0);
    service->addSubService("H", "C", 0);
    service->addSubService("T", "B", 0);

    service->listServices();

    service->listServicesFrom("E");

    service->deleteMainService("D");
    service->listServices();

    agency->addAgency("1");
    agency->addAgency("2");
    agency->addAgency("3");
    agency->addAgency("4");
    agency->addAgency("5");
    agency->addAgency("6");

    agency->listAgencies();

    agency->addOffer("A", "1");
    agency->addOffer("C", "1");
    agency->addOffer("A", "2");
    agency->addOffer("C", "3");
    agency->addOffer("C", "4");
    agency->addOffer("D", "4");
    agency->addOffer("F", "7");

    agency->listServices();

    agency->deleteService("C", "4");
    agency->deleteService("C", "3");
```

```
agency->addOffer("D", "4");
    agency->addOffer("F", "7");

    agency->listServices();

    agency->deleteService("C", "4");
    agency->deleteService("C", "3");
    agency->deleteService("C", "1");

    service->listServices();

    __int64 time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();
    agency->order("E", "1", "phl", 1, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    agency->order("A", "1", "phl", 1, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    agency->order("F", "1", "phl", 1, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    agency->order("B", "1", "phl", 2, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    agency->order("G", "1", "phl", 3, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    agency->order("G", "1", "phl", 1, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    agency->order("G", "1", "phl", 2, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    //Error
    agency->order("B", "1", "phl", 2, time, 1);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();
    //Error
    agency->order("B", "2", "phl", 2, time, 1);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();
    //Error
    agency->order("B", "10", "phl", 2, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();
    //Error
    agency->order("U", "2", "phl", 2, time, 0);
    time = chrono::duration_cast<chrono::milliseconds>(
            chrono::system_clock::now().time_since_epoch()).count();

    agency->order("K", "1", "phl", 3, time, 0);

    agency->listOrder("1");
```

```
agency->listOrder("2");
    return 0;
}
```

Result:

```
C:\Users\Yana\Desktop\MidTermPro\cmake-build-debug\MidTermPro.exe
Service A added successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
Service B added successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
Service C added successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
Service D added successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
The sub service E added to service A successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
Error : The parent of this service is not in list.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
The sub service F added to service A  successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
Error : The parent of this service is not in list.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Error : E is sub service of A .
Can not create loop.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
The sub service B added to service F successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
The sub service K added to service F  successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
Error : The sub service B is already child of its parent.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
The sub service G added to service K successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
Error : The parent of this service is not in list.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Error : G is sub service of A .
Can not create loop.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
The sub service G added to service C successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
The sub service H added to service C  successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
The sub service T added to service B successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - -
A sub services :
E0 (  ) F0 ( B0 ( T0 (  ) ) K0 ( G0 (  ) ) )
```

- - - - - - -
B sub services :

- - - - - - -
C sub services :
G1 ( ) H0 ( )
- - - - - - -
D sub services :

- -
There is not any main service named E .
* *
- - - - - - -
A sub services :
E0 ( ) F0 (B0 (T0 ( )) K0 (G0 ( )))
- - - - - - -
B sub services :

- - - - - - -
C sub services :
G1 ( ) H0 ( )

- -
Agency 1 added successfully.
- -
Agency 2 added successfully.
- -
Agency 3 added successfully.
- -
Agency 4 added successfully.
- -
Agency 5 added successfully.
- -
Agency 6 added successfully.
- -
list of agencies :
1 , 2 , 3 , 4 , 5 , 6 ,
- -
Service A added to agency 1 successfully.
- -
Service C added to agency 1 successfully.
- -
Service A added to agency 2 successfully.
- -
Service C added to agency 3 successfully.
- -
Service C added to agency 4 successfully.
- -
Error : There is no service named D .

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
Error : There is no agency named 7 .
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
- - - - - - -
1 services :
A ---> E0 (  ) F0 ( B0 ( T0 (  )  ) K0 ( G0 (  )  )  )
C ---> G1 (  ) H0 (  )
- - - - - - -
2 services :
A ---> E0 (  ) F0 ( B0 ( T0 (  )  ) K0 ( G0 (  )  )  )
- - - - - - -
3 services :
C ---> G1 (  ) H0 (  )
- - - - - - -
4 services :
C ---> G1 (  ) H0 (  )
- - - - - - -
5 services :
- - - - - - -
6 services :
Service C deleted from agency 4 successfully.
Service C deleted from agency 3 successfully.
Service C deleted from agency 1 successfully.
- - - - - - -
A sub services :
E0 (  ) F0 ( B0 ( T0 (  )  ) K0 ( G0 (  )  )  )
- - - - - - -
B sub services :

- - - - - - -
C sub services :
G1 (  ) H0 (  )

- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Your order added to list of orders of agency successfully.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Your order added to list of orders of agency successfully.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Your order added to list of orders of agency successfully.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Your order added to list of orders of agency successfully.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Your order added to list of orders of agency successfully.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Your order added to list of orders of agency successfully.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Your order added to list of orders of agency successfully.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Error : There is no service named B with input index.
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
Error : There is no service named B with input index.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Error : There is no agency named 10 .
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Error : There is no service named U with input index.
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Your order added to list of orders of agency successfully.
 - - - - - - - - - - - - - - - - - - - - - - - - - - -
Agency 1 orders :
1-
service name : E0 , customer name : phl
priority : 1 , time : 1513539789200
2-
service name : A0 , customer name : phl
priority : 1 , time : 1513539789200
3-
service name : F0 , customer name : phl
priority : 1 , time : 1513539789200
4-
service name : G0 , customer name : phl
priority : 1 , time : 1513539789203
5-
service name : B0 , customer name : phl
priority : 2 , time : 1513539789201
6-
service name : G0 , customer name : phl
priority : 2 , time : 1513539789204
7-
service name : G0 , customer name : phl
priority : 3 , time : 1513539789201
8-
service name : K0 , customer name : phl
priority : 3 , time : 1513539789205
 - - - - - - - - - - - - - - - - - - - - - - - - - - -
Agency 2 orders :
 - - - - - - - - - - - - - - - - - - - - - - - - - - -

Process finished with exit code 0
```

برای چک برنامه و استفاده از testcase های از قبل تعیین شده از TestCase.cpp استفاده کنید .

در غیر این صورت از main.cpp استفاده کنید .

توجه : برنامه با c++11 نوشته شده است . در صورت complie با ورژن های قدیمی mingw ممکن است خطا به وجود آید .

موفق باشید.