گزارش تمرین سری سوم

محبوبه شاکری 9523067

سوال اول)در این سوال برای پیاده سازی Maxheap یک کلاس به این نام تعریف شده و شامل یک متغیر یک پوینتر که ادرس ارایه arr و یک متغیر که size است که اندازه ارایه را نشان می دهد .

(arr به دلیل نیاز در تعریف اپراتور >> به صورت public تعریف شده است)

این کلاس شامل توابع زیر است :

Maxheap() :

تابع Maxheap() که هنگام ساختن یک Maxheap بدون ورودی صدا زده میشود . در این تابع size برابر با صفر قرارداده میشود .

Maxheap(int* , int) :

این تابع هنگامی که میخواهیم یک Maxheap با ورودی یک ارایه بسازیم صدا زده میشود . در این تابع size برابر با سایز ارایه ورودی قرار گرفته و ارایه ای دینامیک به طول ان ساخته شده و اعداد ارایه ورودی به طور متناظر در ارایه arr کپی میشوند. سپس تابع build_max_heap صدا زده شده تا ارایه موجود در Maxheap به صورت مطلوب مرتب شود.

Maxheap(const Maxheap&) :

این تابع هنگامی فراخوانی میشود که بخواهیم یک Maxheap با ورودی یک maxheap دیگر بسازیم و به اصطلاح کپی کنیم . در این تابع سایز Maxheap جدید برابر با ورودی قرار گرفته ، ارایه دینامک متناظر ساخته شده و ارایه maxheap ورودی در ان کپی شده است.

void add(int key); :

در این تابع برای اضافه کردن یک عدد به صورت عمل شده یک ارایه کمکی به نام کپی به اندازه size+1 نیو کرده و اعداد ارایه را فعلی را در ان کپی کرده و در خانه اخر ان عدد ورودی key را قرار داده ، سایز را زیاد می کنیم . ارایه فعلی را دیلیت کرده و مقدار arr را برابر با copy می گذاریم تا همان ارایه جدید را نشان دهد ،

سپس copy را برابر با null قرار داده که در انتها دچار مشکل نشود. سپس تابع add_max_heap را صدا زده تا عدد key به جایگاه مناسب خود برود.

void Delete(); :

در این تابع مشابه تابع add از یک ارایه کمکی استفاده شده و با این تفاوت که سایز ان یکی کمتر از ارایه فعلی است . اعداد ارایه فعلی در جای مناسب در ارایه کمکی قرار گرفته ، سایز maxheap یکی کم شده و ارایه اصلی دیلیت شده و arr به copyاشاره کرده و مقدار copy را null می کنیم .

سپس تابع build_max_heap را صدا زده تا این ارایه جدید را به فرم دلخواه مرتب کند.

int Max(); :

این تابع مقدار ماکزیمم maxheap که خانه صفر ارایه ان است را برمیگرداند.

int getHeight(); :

ارتفاع maxheap از لگاریتم 2 ، size ارایه ان به علاوه یک بدست می اید ، چون خروجی تابع log double است ان را به int، cast کرده تا مقدار مورد نظر بدست اید.

int parent(int); :

برای بدست اوردن شماره خانه parent ابتدا تشخیص میدهیم که عدد ورودی فرد یا زوج است سپس از رابطه مناسب شماره خانه بدست امده و مقدار ان برمی گردد.

int LeftChild(int); :

اگر شماره خانه فرزند سمت چپ یعنی 2n+1 از سایز ارایه کمتر بود یعنی وجود داشت مقدار ان خانه را برمیگرداند .

int RightChild(int);:

اگر شماره خانه فرزند سمت راست یعنی 2n+2 از سایز ارایه کمتر بود یعنی وجود داشت مقدار ان خانه را برمیگرداند .

void printArray(); : در این تابع ارایه maxheap چاپ میشود.

int operator[](int n);: این تابع مقدار خانه به شماره ورودی را برمیگرداند.

void Heapsort(); :

در این تابع در یک حلقه for از خانه صفر تا size هر بار تابع build_max_heap با ادرس خانه i ام و size –I صدا کرده تا در این خانه ماکزیمم i ام قرار بگیرد .

~Maxheap(); :

این تابع هنگام پاک شدن maxheap صدا زده میشود و ارایه را delete می کند.

void max_heapify(int* arr,int length,int i); :

این تابع سه المان ورودی که به ترتیب پوینتر ارایه و طول ان و شماره خانه ای که قرار است بررسی شود .این تابع با این شرط بستگی میشود که اگر خانه I مورد بررسی است فرزندان چپ و راست خود دو maxheap باشند یعنی زیر شاخه ها شرایط maxheap بودن را دارند .

حال این تابع بین خود و فرزندان I بزرگترین عدد را پیدا می کند و اگر بزرگترین در خانه I نبود و یکی از فرزندان بود ان دو خانه را جا به جا کرده و سپس خودش را با ان شماره خانه ای که مقدار خانه I با ان خانه عوض شده صدا میزند زیرا ممکن است با این تغییر شرط maxheap بودن ان شاخه بهم خورده باشد . تا زمانی که همه زیر شاخه ها دوباره مرتب شوند این روند ادامه پیدا می کند.

: void build_max_heap(int*,int);

برای تبدیل ارایه به maxheap از این تابع استفاده میشود و چون در یک ارایه که تبدیل به maxheap میشود نصف اعضا ، دیگر هیچ فرزندی ندارند در نتیجه برای ان ها تابع maxheapify لازم نیست صدا زده شود. پس از عضو با شماره خانه نصف size برای هر کدام تابع maxheapify را صدا کرده و تا خانه اول ادامه می دهیم .

: void add_max_heap(int *, int);

وقتی این تابع صدا زده میشود parent شماره خانه ورودی را با ان مقایسه می کند و اگر لازم به جا به جایی بود ، دوباره همین تابع با شماره خانه parent ورودی صدا زده میشود. و تا زمانی که در جای مناسب قرار بگیرد ادامه پیدا می کند.

: std::ostream& operator<<(std::ostream& os ,Maxheap& a);

خروجی اپراتور >> باید از نوع ostream باشد و اعداد maxheap را به صورتی که به شکل دلخواه ما در بیاید در os ریخته و os را برمیگردانیم.

سوال دوم)در این سوال برای پیاده سازی ساختمان داده vector یک کلاس به نام myVector تعریف شده است که متغیر های ان size , capacity و T* که یک پوینتر است .

برای اینکه با این کلاس بتوان وکتور هایی از جنس های مختلف داشت به صورت class template نوشته شده و شامل myVector.hpp , myVector.h است که در فایل h. تعریف کلاس و در hpp. تعریف توابع نوشته شده است .

Size , capacity از جنس int بوده جون برای وکتور با جنس های مختلف سایز را نشان داده و جنس ان ها ثابت است . اما پوینتری که ارایه وکتور را در بر دارد باید از جنس خود وکتور باشد .

Sizeوکتور با هر اضافه و کم شدن عدد تغییر می کند ، اما capacity به این صورت است که هر گاه size از ان بیشتر شد ، دوبرابر مقدار قبلی میشود .

چون مقدار ان ها نباید از بیرون تغغیر کند به صورت پرایوت تعریف شده و با توابع getsize()
getcapacity (), می توان به مقدار ا ها دسترسی داشت.

حال به تعریف توابع public می پردازیم :

myVector(); :

در هنگام ساخته شدن یک وکتور بدون ورودی صدا زده شده و مقدار size, capacity را برابر با صفر قرار می دهد.

~myVector(); :

در هنگام پاک شدن یک وکتور صدا زده شده و ارایه دینامیک ان را پاک می کند.

myVector(const myVector& myvec) ; :

هنگامی که یک وکتور جدید ازروی یک وکتور دیگر ساخته میشود صدا زده میشود . size , capacity را برابر با وکتور ورودی قرار میدهد و اریه arr را به اندازه new capacity کرده و در یک حلقه از مقدار صفر تا size مقدار ارایه ورودی را در ارایه وکتور جدید کپی می کند.(copy constructor)

: myVector(myVector&&);

هنگامی که یک وکتور جدید از روی یک وکتور دیگر که L value است ساخته میشود ، صدا زده شده و size capacity برابر با وکتور ورودی و arr برابر با arr ورودی قرار گرفته و مقدار arr ورودی را null کرده تا دبل فری رخ ندهد (move constructor)

: void push_back(T);

در این تابع چک میشود که با فزوده شدن یک مقدار دیگر اگر size از capacity بیشتر میشود ، مقدار capcity را دوبرابر کرده و یک ارایه کمکی به اندازه capacity جدید new کرده مقادیر ارایه اصلی را در ان کپی کرده ، عدد ورودی را هم ب ارایه کمکی اضافه کرده ، سپس ارایه وکتور را پاک کرده و پوینتر arr را برابر با copy قرار داده و برای جلوگیری از دبل فری برابر با null قرار میدهد .

اگر هم با افزودن عدد جدید size از capacity بیشتر نشود ، سایز یکی زیاد شده و عدد به وکتور اضافه میشود .

void pop_back();

در این تابع ابتدا یکی از سایزکم شده و اگر size برابر یا کوچکتر از capacity تقسیم بر دو شود لازم است که ارایه وکتور تغییر کند ، در اینجا در روندی مشابه تابع push ارایه وکتور عوض میشود و مقدار capacity هم ابدیت میشود .

void show();

ارایه وکتور را چاپ می کند .

: int getsize();

مقدار size را بر میگرداند

: Int getcapacity();

مقدار capacity را برمیگرداند.

: bool operator<(const myVector&)const;

اگر size وکتور ورودی از size بزرگتر باشد مقدار true و در غیر این صورت false برمیگرداند.

: bool operator==(const myVector&)const;

اگر size وکتور ورودی برابر با size باشد مقدار true و در غیر این صورت false برمیگرداند.

myVector operator+(const myVector&);

این تابع یک وکتور جدید ساخته و با توجه به اینکه size وکتور ورودی یا size بیشتر باشد در ابتدا جمع نظیر به نظیر و سپس ادامه وکتور با طول بشتر را در اریه جدید push کرده و سپس این وکتور جدید را برمیگرداند .

myVector& operator=(const myVector&);

هنگامی که یک وکتور که از قبل ساخته شده برابر یک وکتور دیگر قرار می گیرد . ارایه وکتور پاک شده ، مقدار size , capacity برابر با وکتور ورودی و ارایه ان با طول capacity جدید new شده و ارایه ورودی نظیر به نظیر در ان کپی شده و وکتور برگرداننده می شود .

سوال سوم)

در این سوال ابتدا به تعریف کلاس human می پردازیم .

: Human(std::string a, std::string b ,int ,int ,int, bool c, int);

این تابع constructor این کلاس است و در ان مقادیر ورودی به متغیر های کلاس مقدار دهی می شوند .

bool getGender();

int getHairColor();

int getEyeColor();

int getAge();

void setGender(bool);

void setHairColor(int);

void setEyeColor(int);

void setAge(int);

توابع بالا هر کدام مقدار یک خصوصیت را برمیگرداند یا مقدار ان را عوض می کند .

: Human()~

در هنگام پاک شدن object این کلاس صدا زده و اگر تعداد فرزندان مخالف صفر باشد ارایه ان را پاک می کند .

: bool operator>(Human& a);

اگر سن از سن ورودی بزرگتر باشد مقدار true برمیگرداند.

: bool operator==(Human& a);

تمام خصوصیات را مقایسه کرده و اگر برابر بودند مقدار true برمیگرداند.

: Human& operator++();

سن فرد را افزایش داده و ان را برمیگرداند.

: Human* operator+(Human& a);

اگر ورودی همسر این فرد باشد یک فرزند با سن صفر تولید می کند .و ادرس ان را در ارایه های فرزندان پدر و مادر اضافه می کند .

Database :

1)

در جدول users :

primary key ، برابر با id است . البته چون با هر شماره و تلگرام ای دی فقط یک اکانت می توان ساخت این دو ردیف هم می نوانستند به عنوان primary key انتخاب شوند .

در جدول blockuser :

Primary key دو ستون blocked_user_id , blocker_user_id هستند ، زیرا یا یکی از ان ها یک ردیف متمایز مشخص نمیشود بلکه هر ذو با هم می توانند یک ستون متمایز را مشخص کنند (یک نفر چند نفر را بلاک کرده و یک نفر ممکن است توسط چند نفرربلاک شده باشد)

Foreign key هم همین دو ستون هستند که id ها بلاک کننده و بلاک شونده باید در جدول users موجود باشد .

در جدول messages :

Primary key برابر با id و foreign key ها ستون های sender_id , receiver_id هستند که باید در جدول users موجود باشند.

در جدول channel :

Primary key برابر با id است (ستون telegram_id هم میتوانست باشد)

Foreign key برابر با creator _id است که باید در جدول users موجود باشد.

در جدول groups :

Primary key برابر با id است و creator_id به عنوان foreign key باید در جدول users موجود باشد.

در جدول groupmessage :

Primary key برابر با id و sender_id به عنوان foreign key باید در جدول users موجود باشد . و Group_id به عنوان foreign key باید در جدول group موجود باشد.

در جدول channelmassage :

Primary key برابر با id و channel_id به عنوان foreign key باید در جدول channel موجود باشد.

در جدول messageattachment :

Primary key, foreign key در این جدول message_id است که باید در جدول message موجود باشد.

در جدول groupmessageattachment :

Foreign key ,primary key برابر با message_id است که باید در جدول groupmessage موجود باشد .

در جدول channelmessageattachment :

Foreign key ,primary key برابر با message_id است که باید در جدول channelmessage موجود باشد.

در جدول channelmembership:

دو ستون user_id , channel_id هر دو با هم primary key هستند و همین ستون ها foreign key هم هستند که باید در جدول channel, users موجود باشند.

در جدول groupmembership :

دو ستون user_id , group_id هر دو با هم primary key هستند و همین ستون ها foreign key هم هستند که باید در جدول users, group موجود باشند.

2) داده های مورد نیاز برای query های مد نظر به database اضافه شده و در فایل insert.sql موجود است.

3) Query ها در فایل query.sql موجود است و اسکرین شات نتایج هم با شماره query در فایل قرار گرفته است.
4) برای اینکه بتوان در کانال ها چند نفر ادمین داشت باید یک جدول دیگر به نام channeladmins داشت که ستون های admin_id و channel_id داشته باشند که هر کدام به جدول های users , channel به عنوان foreign key مربوط شده و در جدول channelmassege ستونی با نام sender_id موجود باشد که به عنوان forigne key به همراه channel_id چک شوند که ستونی یا این دو id در channeladmins موجود باشد .
برای گروه هم جدولی با نام groupadmin باید داشته باشیم که ستون های admin_id و group_id به عنوان foreign key در جدول users , group موجود باشند.

تمرینات در ادرس https://github.com/MahboobeSh/AP-HW3 قرار داده شده اند.