پروژه میانی درس برنامه نویسی پیشرفته دکتر جهانشاهی_ نیلا مسروری سعادت_۹۵۲۳۱۱۱

در ابتدا برای ساخت کلاسی که مکعب های روبیک را درست کند، ازکلاس های Row و Side استفاده کردم.کلاسی که مکعب کیوب را می سازد و تمام عملیات روی کیوب را در ان انجام می دهدکلاس cube نام دارد. هم چنین سایرکدها و اجرا و پیاده سازی الگوریتم DLSو IDS برای حل مکعب روبیک و ارتباط با کاربر در main.cpp صورت گرفته است.

کلاسRow :

برای درست کردن هر ردیف روی هر وجه مکعب می باشدکه هر ردیف هم دو cubie(کیوبی) دارد با نام های left و right و از جنس int

```
Row();//default constructor
```

Default constructor:

در این تابع کانستراکتور یک ردیف با مقادیر صفر می گذاریم و هنگام ساختن یک شی خالی ازکلاس صدا زده می شود.

```
Row(const Row &r);//copy constructor
```

Copy constructor:

این تابع زمانی صدا زده می شودکه بخواهیم یک شی جدید ازکلاس بسازیم و شی تعریف شده ی قبلی را در ان کپی کنیم . برای این کارکافیست هرکیوبی چپ و راست ردیف شی ساخته شده را درکیوبی چپ و راست شی جدیدکپی کنیم.

```
Row(int l, int r);
```

تابع کانستراکتور اصلی کلاس که هر شی را می توان با داشتن دوکیوبی اش ساخت.

```
int getleft() const;//left cubie
int getright() const;
void setleft(int c);
void setright(int c);
```

توابع setter getter برای دستیابی و یا عوض کردن مقادیر leftو right که متغیر های private کلاس می باشند.

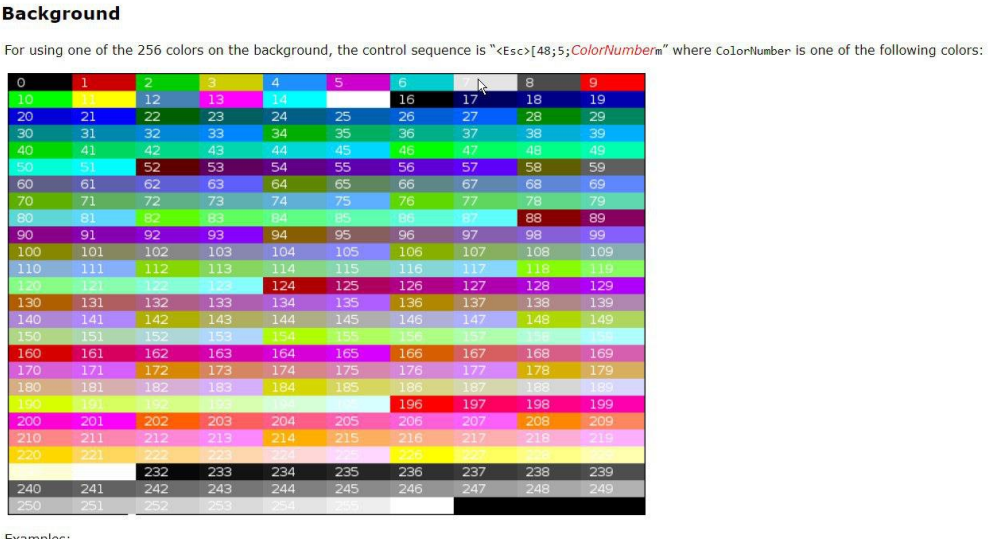
```
void check_color();
```

این تابع صرفا برای رنگی کردن اعداد مختص هرکیوبی با استفاده از ANSI escape code می باشد و بعدا درکلاسcube برای رنگی چاپ کردن ان در تابعshow_colored استفاده می شود به این صورت که وقتی روی هر شی ازکلاس row صدا زده شود هرکیوبی ان ردیف را چک می کند و متناسب با عدد ان ،ان را با رنگ متناظر چاپ می کند.

برای رنگی چاپ کردن هر چیزی با استفاده از ANSI escape code از دستورکلی زیر استفاده می شود:

```
cout<<"\e[48;5;colornumber m"<<what you want to be colored<<"\e[0m";
```

که colornumber را می توان از جدول زیر پیدا کرد.




```
Row flip();
```

این تابع برای جابه جا کردن دوکیوبی چپ و راست هر ردیف می باشد و بعدا برای چرخش ها و جابه جایی ردیف ها درکلاس cube استفاده می شود.

```
bool same();
```

این برای چک کردن تساوی کیوبی چپ و راست می باشد. در صورت برابر بودن true ریترن می شود.

بعدا از ان در تابع issolvedکه معادل همان goal_test می باشد استفاده شده است.

```
int color;
```

این یک متغیر برای نسبت دادن رنگ کیوبی ها به وسیله عدد است و اگر رنگ یکسان داشته باشند مقدار ۱- و در غیر این صورت مقدار color id هرکدام را می گیرد.

کلاس Side :

این کلاس برای تعریف هر وجه می باشد و شامل متغیر هایی از جنس کلاس row می باشد به اینصورت که هر وجه دو ردیف upper و lower دارد که به صورت private هم تعریف شده اند.

```
Side();//default constructor
```

Default constructor:

این تابع دیفالت کانستراکتور این کلاس می باشد و برای ساختن یک شی خالی از این کلاس است که در ان فقط مقدار color را برابر ۱- قرار می دهیم که این به این معناست که کیوبی های چپ و راست هر دو ردیف بالا و پایین برابر است.

```
bool same();
```

مشابه ان در کلاس row هم بود با همان کارایی با این تفاوت که هر۴ کیوبی هر وجه را چک می کندکه برابر یا همرنگ هستند یا نه و اگر بودند مقدار true را ریترن می کند.

```
void CW(); // performs clockwise turn on this side
    void CCW(); // performs counterclockwise turn on this side
```

برای چرخش ساعتگرد و یا پاد ساعتگرد هر side می باشد وکافیست که در ان کیوبی ها را به صورت ساعتگرد بچرخانیم. مثلا

```
/* Orientation
    | a b |
    | c d |
    */
```

این هر وجه به صورت اصلی و قبل چرخش می باشد.

بعد چرخش cw :

```
/* Orientation
    | c a |
    | d b |
    */
```

برای چرخش پادساعتگرد هم به همین صورت ولی برعکسش می باشد.

```
void makeBlank(); // creates a blank side with zero cubies
```

این تابع یک وجه با کیوبی هایی همه با مقدار صفر و بدون رنگ را درست می کند.و بعدا از این تابع برای راحت تر شدن کار در تابع createempty درکلاس cubeاستفاده می شود.

```
    Row getupper() const;
    Row getlower() const;
    void setupper(Row r);
    void setlower(Row r);
```

توابع getterو setter برای دستیابی و یا حتی عوض کردن متغیر های private کلاس می باشد.

```
    Row getleftside() const;
    Row getrightside() const;
    void setleftside(Row r);
    void setrightside(Row r);
```

این ها هم برای گرفتن و یا ست کردن rightside و leftsideهستند .

Rightside در واقع همان ستون عمودی هر side شامل کیوبی سمت چپ ردیف بالا و پایین می باشد.

Leftside در واقع همان ستون عمودی هر side شامل کیوبی سمت راست ردیف بالا و پایین می باشد.

کلاس cube :

این کلاس ازکلاس های side و rowاستفاده می کند برای تعریف کردن هر شی مکعب یا کیوب.

متغیر های private کلاس :

```
Side up, right, left, front, back, down;
```

شامل متغیر هایی از جنس کلاس side میباشند یعنی هر مکعب داری ۶ وجه با نام های up,right,left,down,back,front می باشد.

```
void createEmpty(); // initializes an empty cube
```

یک مکعب خالی می سازد که همه ی کیوبی های هر وجه ان برابر صفر می باشد.

```
Cube();//default constructor
```

Default constructor

این تابع کانستراکتور دیفالت کلاس می باشد و یک کیوب خالی با استفاده از تابع createEmpty می سازد.

```
Cube(const Cube &c);//copy constructor
```

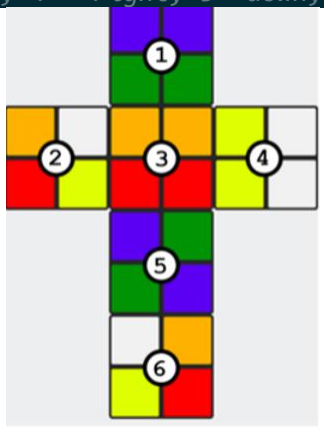
این تابع کانستراکتور اصلی این کلاس می باشد.که در ان هر وجه مکعب با استفاده از تابع getside که روی هر شی مکعب تعریف شده است ، مقدار دهی می شود. در زیر به برسی تابع getside می پردازم.

```
Side getSide(int a) const;
```

این تابع با گرفتن ورودی عددی به صورت int معادل وجه ان عدد را بر می گرداند به صورت شی ای از side .

اختصاص هر وجه به یک عدد به صورت زیر می باشد.

```
/* 1 = up, 2 = left, 3 = front, 4 = right, 5 =down, 6 = back */
```



```
void set(int side, int a, int b, int c, int d);//setting the numbers of each side
```

این تابع خیلی برای گرفتن اعداد هر وجه (کوبی ها) و ساختن اون وجه به کار می رود.

برای این کار شماره وجهی که می خواهیم بسازیم و اعداد و در واقع کیوبی های اون وجه را وارد می کنیم.

اول در این تابع ردیف ها توسط توابع setupperو setlower و اعداد این کیوبی ها ساخته می شود.و بعد به هر وجه مورد نظر اختصاص داده می شود.

```
void num_assign();
```

این تابع برای تست ها بسیارکاربردی است و با استفاده از ان در واقع کیوب مورد نظرکه این تابع رویش صدا زده شده را به ضورت زیر می سازد.

```
# Rubik's cube internal indexing is
#                +-------+
#                | 0   1 |
#                | 2   3 |
#        +-------+-------+-------+
#        | 4   5 | 8   9 | 12 13 |
#        | 6   7 | 10 11 | 14 15 |
#        +-------+-------+-------+
#                | 16 17 |
#                | 18 19 |
#                +-------+
#                | 20 21 |
#                | 22 23 |
#                +-------+
```

```
void show();//showing the cube with numbers
```

این تابع برای نمایش کیوب با اعداد هرکیوبی و به صورت عکس بالا می باشد .ترتیب وجه ها هم در بالاتر ذکر شده است.برای دسترسی به اعدادکیوبی هر وجه ابتدا از توابعgetlower, getupperبرای دستیابی به سطر هر وجه و از توابع getright, getleft برای دستیابی به کیوبی سمت چپ و راست هر سطر استفاده شده است.

```
char num2char(int a)
```

این تابع که از توابع کلاس نیست برای این منظور استفاده می شود که حرف رنگ معادل هر عدد را برگرداندکه اختصاص این charها به صورت زیر می باشد.

```
/* Color notation:
 * 1: orange
 * 2: green
 * 3: white
 * 4: blue
 * 5: red
 * 6: yellow
 */
```

```
void displayCube();//showing the colors with char
```

این تابع دقیقا مانند تابع برای نمایش ماتریس می باشد و البته این بار قبل از چاپ هر عددکیوبی روی هر وجه انرا توسط تابع num2char به char رنگ متناظرش تبدیل و ان را چاپ می کند.

```
void show_colored();//showing the numbers in a colored form
```

این تابع یک فرم دیگر از نمایش کیوب می باشد.که در ان از تابع checkcolorکه قبلا توضیح ان داده شده ، استفاده شده است.که همون تابع showمی باشد ولی به صورت رنگی و با رنگی که به هر عدد اختصاص می یابد.

```
bool isValid();
```

این تابع برای اطمینان از صحت هرکیوبی است که درست می شود از این بابت که مطمین شویم که از هر رنگ یا به قولی از هر عددی فقط ۴ عدد وجود دارد.و اگرکیوب درست باشد از این لحاظ true را برمی گرداند.

```
bool isSolved();//equal to the goal_test
```

این تابع معادل همان تابعgoal_test ای می باشدکه در pdf راهنمایی پروژه توضیح داده شده است.

در این تابع به بررسی این می پردازد که اولا کیوب خالی نباشد و ثانیا اگر هر وجه کیوب دارای اعداد یکسانی است یعنی کیوب حل شده و مکعب روبیک ما کامل شده است.

```
void U(); // U clockwise
    void Up(); // U counterclockwise
void R();
    void Rp();
void L();
    void Lp();
void F();
    void Fp();
void B();
    void Bp();
void D();
    void Dp();
```

این توابع برای جایگشت و چرخش هر وجه هستندکه بر روی اعداد وجه های دیگر هم تاثیر می گذارند. برای مثال با چرخاندن وجه up به صورت ساعتگرد تابع u صدا زده می شود. در این تابع علاوه بر این که خود وجهup باید با صدا زدن تابعcw() چرخانده شود، سطر هایی از همه وجه های دیگردر مجاورت ان جابه حا می شوند. بعضی اوقات در جابه جایی سطر ها ، باید جای المان چپ و راست هم جابه جا شود که این کار با فراخوانی تابعflipروی هر سطر انجام می شود. بعد از انجام هرکدام از چرخش ها نوع چرخش هم چاپ می شود که برای ساعتگرد همان حرف و برای پادساعتگرد همان حرف به همراه ' می باشد مثل 'u.

```
Cube Rotate(int side,const string r);
```

این تابع مطابق توضیح داده شده در pdf مربوط به پروژه می باشد.با گرفتن side به صورت عدد مربوط به اون وجه ، و هم چنین نوع چرخش به صورت رشته کیوب را می چرخاند و کیوب چرخانده شده را تحویل می دهد . در این تابع از همان توابع U,Up,R,Rp,...که در بالا توضیح داده شده ، استفاده شده است.

```
int depth;//for DLS
```

این متغیرکه به صورتpublic تعریف شده است، برای تخصیص دادن عمق به هر یک ازکیوب ها در نود های گراف الگوریتم dls می باشدکه بعدا از ان در حل روبیک توسط این الگوریتم استفاده می شود.

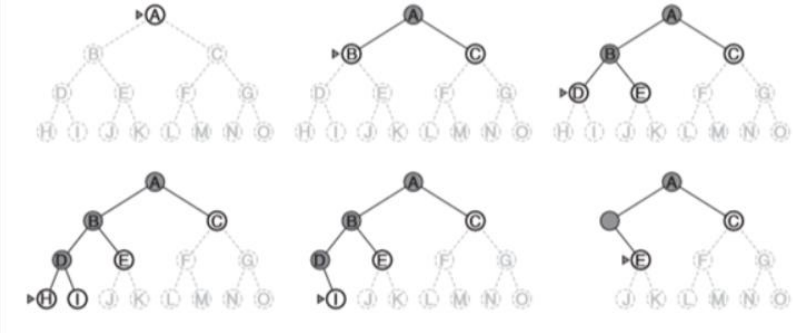
این یعنی اینکه هرکیوب در ان لحظه دارای چه عمقی می باشد.از ۰ تا عددlimit

پروژه میانی درس برنامه نویسی پیشرفته دکتر جهانشاهی_ نیلا مسروری سعادت_۹۵۲۳۱۱۱

Main.cpp:

توضیح الگوریتم depth_first_searchو مقایسه ان با الگوریتمdepth_limited_search :

این الگوریتمی برای جستجوی ساختار داده های درخت یا نمودار است. این تکنیکی است که به طور گسترده ای برای ایجاد راه حل برای حل مشکلات در هوش مصنوعی کاربرد دارد. این روش همچنین به عنوان یک تکنیک قدرتمند برای حل مشکلات گرافیکی مختلف و همچنین برای ماز مارها به رسمیت شناخته شده است. فرض کنید یک نمودار داده شده است. با شروع از یک راس G ، الگوریتم در امتداد لبه ها حرکت می کند ، از راس گرفته تا راس تا رسیدن به یک عمق برش .هنگامی که به حد ممکن برسد (لبه ها) ، آنگاه به گره اخیر اخیراً گسترش یافته بازگشت و اکتشاف جدید را از این نقطه شروع می کند. سرانجام ، این الگوریتم در تمام لبه های G ، هر یک دقیقاً یک بار ، طی می شود. تنها مسیری که به منظور اجرای الگوریتم ذخیره می شود ، مسیر بین گره اولیه وگره فعلی است. از آنجا که الگوریتم جستجوی اول عمق تنها مسیر بین گره اولیه و مسیر فعلی را ذخیره می کند ، موظف است تمام مسیرهای موجود در نمودار را تا عمق برش جستجوکند. نخستین جستجوی عمق یک الگوریتمی است که حداقل نیاز به فضای کمتری دارد و از اهمیت بالایی برخوردار است. می توان اولین الگوریتم جستجوی عمق را با یک پشته اولین بار در خارج اجرا کرد یا به عنوان یک اجرای بازگشتی پیاده سازی کرد. الزامات حافظه این الگوریتم در حداکثر عمق جستجو به صورت خطی است ، زیرا الگوریتم فقط برای ذخیره حالت هایی که در حال حاضر در پشته جستجو هستند نیاز دارد. یکی دیگر از مزیت های این الگوریتم این است که اگر بدون کاوش در کل درخت راه حلی پیدا کند ، زمان و مکانی که می گیرد کمتر خواهد بود. با این حال ، یک الگوریتم جستجوی اول عمق دارای اشکالات مختلفی است. با توجه به نتیجه گیری مکعب ، تعداد حرکات مشخص وجود دارد. اگرکسی بخواهد راه حل مکعب روبیک را پیدا کند ، باید در نظر بگیرد که کل فضای فعلی برای این مشکل بسیار زیاد است ، که ممکن است باعث شود الگوریتم لبه ای را انتخاب کندکه منجر به یک زیرگرافی شود که حاوی حالت راه حل نیست و عظیم است. به طورکامل گسترش یابد این نتیجه ای به راه حلی می دهدکه هرگز توسط این الگوریتم بازگردانده نمی شود ، زیرا کد ممکن است همان صفحه را به صورت متناوب به عقب و عقب بچرخاند و پیشرفت آن را برای برنامه غیرممکن می کند.
برای غلبه بر مشکل این الگوریتم جستجوی عمق ، محدودیت های هرس ، depth_limited_search اضافه می شود. هرس متحرک نمونه ای از هرس است که از تکنیک کم سربار برای کاهش اندازه درخت اول جستجوی عمق استفاده می کند و باعث می شود راه حل های مناسب و همچنین عملکرد بهتری انجام شود. با محدودکردن تعداد لبه ها / عمق چند الگوریتم تا زمان بازگشت به جستجوی آن ، همیشه اگر در عمق معین وجود داشته باشد ، راه حل ارائه می دهد. این به عنوان جستجوی محدود عمق شناخته می شود.



```
Cube depth_limited_search(Cube c,int limit,int main_limit,std::vector<Cube> stack
);
```

این تابع کیوب و عمق و یک وکتوری از جنس cube را می گیرد. در این تابع نود ها را که در وکتور ذخیره می کنیم روش بیرون کشیدنشان به صورت last in first out می باشد.

در هر مرحله تا زمانی که iter از limit کمتر باشد، می اید و childrenهای اخرین cube ذخیره شده در stackرا در استک یا همان وکتور push می کند .حالا بچه های هرکیوب یا هر نود که در استک هست به عنوان والد در واقع انواع چرخش های هر slide ان والده.

این چرخش ها توی حلقه هست که این حلقه وجه های این مکعب را به ترتیب و به صورت cw و ccw می چرخاند.و هر بار بعد از این کار به عدد iter که در واقع شمارنده عمق هست اضافه می کند .هم چنین لازم است که عمق هر نود و هر بچه جدید را در depth ان کیوب ذخیره کنیم . هر child عمقش یک عدد از عمقparentاش بیشتر است.

در صورتی که iter از limitبیشتر شود به این معناست که به عمق اخر رسیده است.

و وارد else میشود در else می اید تمام ۱۲ بچه های اخر نود عمق یکی مونده به اخر را تک تک چک می کند و pop_back می کند.و دوباره همین تابع را با عمق cntصدا می زند.

نکته اصلی بازگشت در این جا متغیر cnt می باشدکه عمق جدید را تعیین می کند.

یعنی هنگامی که می خواهد به عمق کمتر بازگردد، باید چک کنیم که عمق parent یا کیوب(نود)کنونی چه مقداری دارد و انرا از main_limit که همون مقدار limit اولیه است(و باید حتما ثابت باشد) کم می کنیم تا cnt بدست بیاید.

علت این کار این است که بفهمیم که نود کنونی با عمق اخر چند عمق یا چند مر حله فاصله دارد و بتوانیم دوباره با این عمق جدید و با عنوان limit جدید تابع را دوباره صدا بزنیم.

*دقت شود که main_limitباید در تمام مدت ثابت بماند.

*قبل از فراخوانی این تابع باید حتما vectorای از جنسcube ساخته شود و حتما cube مورد نظر به عنوان اولین المان داخل ان push_back شود.

```
bool dls(Cube c,int limit);//my second approach for solving with dls method
```

این یک روش دیگر برای اجرای dlsمی باشدکه مدت زمان بیشتری برای رسیدن به جواب طول می کشد.

در اینجا با یک حلقه ابتدا همه اسلاید ها را به صورتcw و سپس ccw می چرخاند.و بعد از هر چرخش در واقع ان نود را با صدا زدن همین تابع به صورت بازگشتی و با یک عدد عمق کمتر،گسترش می دهد.

خروجی این تابع به صورت boolian می باشد.

```
Cube  depth_limited_search_increment(Cube c, int max_depth=10);//IDS
```

این تابع برای اجرای الگوریتمIDSمی باشد.

و این تابع در واقع تعیین می کندکه DLS با چه عمقی اجرا شود مثلا ممکن است ماکزیمم عمق ۱۰ باشد ولی این کیوب تا عمق ۴ حل شود و لازم به طی کردن کل ۱۰ عمق نباشد.و با یک حلقه هر بار از عمق ۰ تا ماکسیمم عمق تابع depth_limited_search را صدا زده وکیوب تولید شده را چک می کند تا به کمترین عمق مناسب برای حل این کیوب برسد.

دریافت کیوب ازکاربر:

برای این در قسمت main یک ارایه و یک حلقه تعریف شده است و برای هر وجه با نامsurface اعداد ان وجه را ازکاربر می گیرد .و ۴ تا ۴ تا در ارایه برای هر وجه ذخیره می کند و در حلقه اصلی که

شمارنده وجه ها می باشد ان وجه مورد نظر i را با تابعset و اعداد وارد و ذخیره شده در ارایه می سازد در نهایت کیوب با داشتن همه وجه ها ساخته می شود.

در ضمن اعداد هر وجه را کاربر باید با زدن space ازهم وارد کند.

```
int arr[4];//for getting the cubies of each surface

    for (size_t i = 1; i <7 ; i++)
    {
        std::cout<<"Surface "<<i<<": ";
        for (size_t j = 0; j < 4; j++)
        {
            std::cin>>arr[j];
        }
        cube.set(i,arr[0],arr[1],arr[2],arr[3]);
}
```

در این جا به تست و بررسی یک نمونه می پردازم:

```
#test1
1,1,1,1
6,6,2,2
2,2,3,3
3,3,4,4
5,5,5,5
6,6,4,4
```

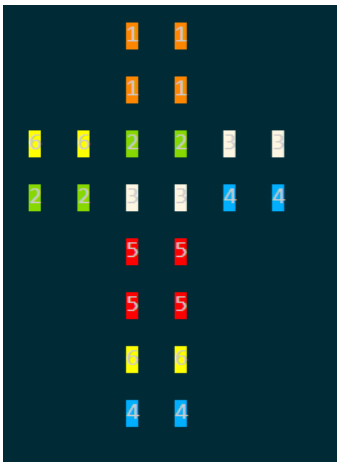
بعد از نمایش با تابع show:

```
      1  1
      1  1
6  6  2  2  3  3
2  2  3  3  4  4
      5  5
      5  5
      6  6
      4  4
```

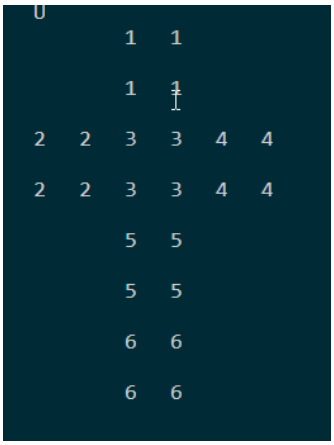
بعد از نمایش با تابع displaycube:

```
         o  o
         o  o
y  y  g  g  w  w
g  g  w  w  b  b
         r  r
         r  r
         y  y
         b  b
```

بعد از نمایش با تابع show_colored:



بعد از چرخاندن وجه ۱ کیوب و نمایش ان:



بعد از اجرای این خطوط و حل ان با dls: