توضیحِ روشِ حل مسئله‌ی Dining Philosophers به کمکِ الگوریتمِ بانکدار

تهیه و تنظیم: مبین خیبری

شماره دانشجویی: 994421017

استاد راهنما: دکتر لیلا شریفی

برای اجرای درستِ این برنامه لازم است که به پوشه‌ای که فایل‌های مربوط به برنامه در آن قرار دارند، تغییرِ مسیر داده و سپس برنامه را با دستورِ زیر کامپایل کنیم:

```
make banker
```

الگوریتمِ بانکدار سعی می‌کند که سیستم را در حالت امن نگه دارد. به همین دلیل هر درخواستی که به الگوریتم فرستاده می‌شود، یا پذیرفته خواهد شد (Accepted) و یا رد خوهد شد (Rejected).

برنامه‌ی فوق برای اجرا دست کم به دو نوع آرگومان نیاز دارد:

Available Matrix که آن را مستقیما از کاربر دریافت کرده و Max Matrix که مقادیرِ آن را از قسمتِ MAX_FILE_NAME خواهد خواند.

به عنوان مثال، Available Matrix در دستورِ زیر [5 5 6 6 5] خواهد بود.

یک نمونه از اجرای برنامه:

```
./banker.out 5 6 6 5 5
```

پارامترهای اصلیِ مورد نیاز برای اجرای این برنامه در زیر به طور مفصل معرفی شده‌اند:

NUMBER_OF_RESOURCES: میزانِ منابع را نشان می‌دهد. برای حل مسئله، فرض می‌کنیم این مقدار، نشان‌دهنده‌ی تعداد چنگال‌های موجود است.

NUMBER_OF_CUSTOMERS: میزانِ مشتری‌ها و یا متقاضیانِ استفاده از منابع را نشان می‌دهد. در اینجا فرض می‌کنیم این عدد بیانگرِ تعداد فیلسوف‌هاست.

MAX_CUSTOMER_ITERATION: بیشینه‌ی تعدادِ ارسال درخواست به الگوریتم بانکدار توسط هر فیلسوف، به کمکِ این عدد مشخص خواهد شد.

الگوریتم بانکدار به ما کمک می‌کند که توزیعِ منابع را به شکلی انجام دهیم که سیستم هیچگاه به بن‌بست نرسد. در واقع هیچ فیلسوفی تا ابد در انتظار غذا نخواهد ماند.

اجرای برنامه:

همانطور که در بالا اشاره شد، در این برنامه بردارِ Available در ابتدای کار توسط آرگومان‌های برنامه گرفته می‌شود و ماتریسِ Maximum از فایلِ max.txt که در دایرکتوری برنامه موجود است خوانده می‌شود. نمونه خروجیِ برنامه با دستوراتِ زیر در ادامه آورده شده:

```
make banker
```

```
./banker.out 5 6 6 5 5
```