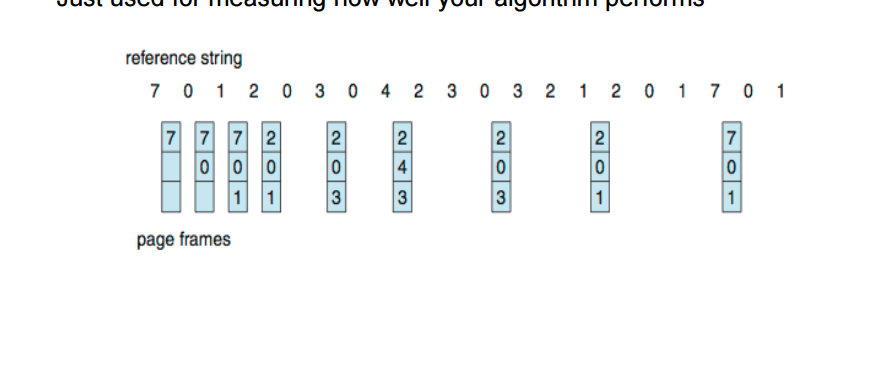
컴퓨터학부 20201853 서민비

1. 개요

일곱 가지 페이지 교체 알고리즘(Optimal, FIFO, LIFO, LRU, LFU, SC, ESC)을 C언어로 구현한다. 그리고 시스템에서 자동(랜덤)으로 입력 또는 사용자 파일 입력된 참조 페이지 스트링을 통해 페이지 교체 알고리즘 각각의 페이지 폴트 횟수를 측정하고 Optimal 알고리즘과 비교하여 성능을 알아본다. 단, 참조 페이지 스트링은 1~30까지 있다고 가정하며, 최소 500개의 페이지 스트링이 존재한다. 페이지 프레임의 개수는 사용자 입력으로 최소 3에서 최대 10개의 프레임이 존재할 수 있다. 시뮬레이션 결과가 표준 출력이 되고 동시에 파일(result.txt)로도 저장하는 기능이 제공된다.

* 구현한 페이지 교체 알고리즘 동작 방식 설명 //참고문헌 : 김철홍 교수님 운영체제(라) 강의 자료 및 구두 설명
  + Optimal

가장 오랫동안 사용하지 않을 페이지를 교체한다. Page fault가 발생하면, 모든 페이지가 이후에 사용되는 지 여부와 이후에 등장할 때의 index를 검사한다. 페이지 중 이후에 등장하지 않는 페이지가 있다면 해당 페이지를 교체한다. 모든 페이지가 이후에 다시 등장한다면 가장 오랫동안 사용하지 않는 페이지(index가 가장 먼 페이지)를 교체한다. 우선적으로 교체할 조건에 부합하는 페이지가 2개 이상이 존재한다면 가장 낮은 프레임 번호를 가진 페이지를 교체한다.



참고문헌 : 김철홍 교수님 운영체제(라) 강의 자료 및 구두 설명

* + FIFO

가장 오래 전에 들어온 페이지를 우선적으로 교체한다. FIFO Queue로 구현하였으며 queue가 full 상태가 되기 전에는 queue에 페이지를 순서대로 저장하다가 full 상태가 되면 queue에 가장 먼저 들어온 페이지를 교체한다.

테이블이(가) 표시된 사진

자동 생성된 설명

참고문헌 : 김철홍 교수님 운영체제(라) 강의 자료 및 구두 설명

* + LIFO

가장 최근에 들어온 페이지를 교체한다. Stack으로 구현하였으며 stack이 full 상태가 되기 전에는 stack[top++]하며 페이지를 불러들이다가 full 상태가 되면 stack[top]에 존재하는 페이지를 교체한다.

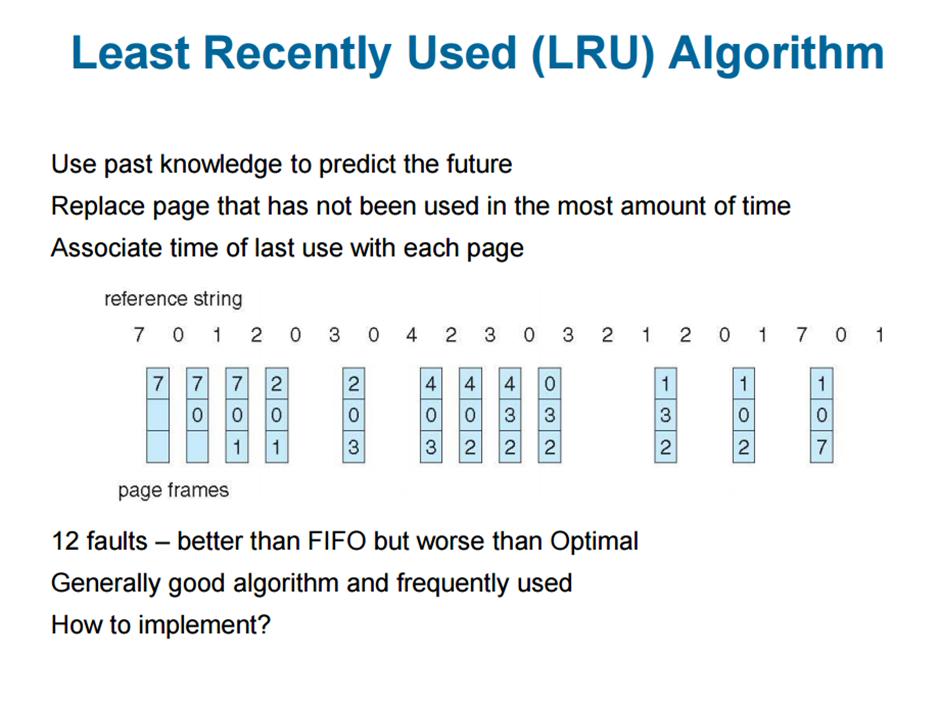
테이블이(가) 표시된 사진

자동 생성된 설명

참고문헌 : <https://www.scaler.com/topics/operating-system/page-replacement-algorithm/>

* + LRU

가장 오랫동안 참조하지/사용되지 않은 페이지를 교체한다. 각 페이지는 참조되거나 사용될 때마다 시간이 int 값으로 기록된다. 가장 최근에 참조/사용된 페이지는 가장 큰 시간 값을 가지게 된다. 따라서 시간이 가장 오래된(시간 값이 가장 작은) 페이지를 찾아 교체한다. 우선적으로 교체할 조건에 부합하는 페이지가 2개 이상이 존재한다면 가장 낮은 프레임 번호를 가진 페이지를 교체한다.



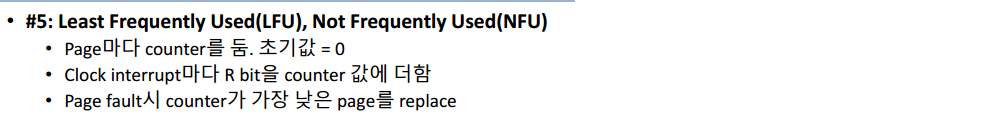
텍스트, 실내, 스크린샷이(가) 표시된 사진

자동 생성된 설명

참고문헌 : 김철홍 교수님 운영체제(라) 강의 자료 및 구두 설명, [LRU Algorithm | Least Recently Used Page Replacement Algorithm - YouTube](https://www.youtube.com/watch?v=CMsn2-diunc)

* + LFU

페이지 참조 횟수가 가장 작은 페이지를 교체한다. 각 페이지는 참조 횟수를 계산하는 counter값을 가진다. 따라서 한번 참조될 때마다 해당 페이지의 counter값은 1씩 증가한다. Page fault가 발생하면 counter 값이 가장 작은 페이지가 교체된다. 우선적으로 교체할 조건에 부합하는 페이지가 2개 이상이 존재한다면 가장 낮은 프레임 번호를 가진 페이지를 교체한다. //결과 출력 시 (counter 값)도 함께 출력했습니다.



**참고문헌 : 김철홍 교수님 운영체제(라) 강의 자료 및 구두 설명**

* + SC

참조 비트가 1이면 한 번 더 기회를 주고, 참조 비트가 0이면 해당 페이지를 교체한다. circular queue로 구현했으며 pointer는 가장 오래된 page를 가리킨다. Page fault가 발생하면, 포인터가 가리키는 페이지의 참조 비트를 검사하여 참조 비트가 1이면 참조 비트를 0으로 변경 후 포인터는 다음 페이지를 가리킨다. 포인터가 가리키는 페이지의 참조비트가 0이라면 해당 페이지는 교체되고 포인터는 다음 페이지를 가리킨다.

![텍스트, 다른, 묶음, 여러개이(가) 표시된 사진

자동 생성된 설명](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RCURXhpZgAATU0AKgAAAAgABAE7AAIAAAALAAAISodpAAQAAAABAAAIVpydAAEAAAAWAAAQduocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEFkaXR5YSBQYWwAAAAB6hwABwAACAwAAAhoAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBAGQAaQB0AHkAYQAgAFAAYQBsAAAA/+EKY2h0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkFkaXR5YSBQYWw8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgBvAO8AwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kaKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigCjY6kb2+vbY2dxB9kcIJJUwsuR1Q9xV6iigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOe1Pxxouk6pJp109y1zEoZ0htnkwD05UGtLSNb0/XbU3GmXCzKDhl6Mh9GB5B+tcHceJLTw38VdZlvYbqVZrWFVFvEZCCPXHStPwnZXt3ruteIo7ZtOt9QVVt4ZxhiVH32UcDmuWnWcn33+Vm/zPdr4ClToc9mvdi021Zt2ukrX6vvsdxRXmHiHW9Q8NWf25fGS393C6tJY+VHskUkBgMDPrW54g1q/l1/RNNtdQXS7e/haaWcqCxwAQik8A81oqqa2/I5nllT3WpKzvrr9lXe6vt2R2dFcD4YvtRPi3W4bvX5tRsdMRWX5ECnIJIJA6jHas608UT63bXOozeLY9JdmcW1kkasECnALEjJzj9an28bX/rTQr+y6vM0pKytraX2ldaWvt5Hp9FYHgnXpfEnhO11G5UCZtySYGAWU4JHscVgahrN7P4/u9Ju9ck0a2jiQ2oWJcTEgZO5h+laymo28zCGBqSqzpPRwvfd7O3RM76iuQ8X67d+H9EsLa3v4xe3cqxfbZlG1R1ZyBx0rJg8Rz6f4k0i3tvEq61bXsvkXEbxqHRiDhgQBxmpdWKly/IulltWrT9pF6a231tv0svK9j0WiuFtbnxB4q13VjY6udLstPmNvAkcSuZXHUtkdKZpGt622geLG1O7V7zTXlSJ41G1CsWRjj155pe2Vr28xvLppP31dWutbrmtbpbrrY72iuRuNYvk+ErastwRfDTfO87Azv25zjpVLxB4qubSw0Kxiv47K51KJXmvpFB8pQuSQOmSeKcqsYuz8vxJp5fVqS5Y23a/8BV2/T8fI7uiuC0fxFJB4ystMi8Qf23aXsMmS0ahoXQZ6gDgjNUvDUnirxSmqv8A8JC1lBa300MRjhRnJB4ByPugfjS9snsv6Ro8snFOU5pJJO7v1bW1r7p9D0qiuBh8ZahH8O5b2XZJqa3LWEcm0bXlDbQxH4ZrW0/R/ENm9td3niR7pQN1zA8CBSMZwpAyKpVE3ov6ZlPAypJ+0klZtLfVrtZfnY6iivMrTxTLrrXd7N4sTR0Ejx2tokatgKcBmJHOcVraNqut+Mfh552mXsdlqolMRuSnykqeSBg8EVMaymtPU1q5XVpK82krpPfS/wAtfO1zt6Ky9b1GXRPCt5fviWW0ti5P95gv+NcZJpE8XghPFf8AaV7/AGulsL1285vLc43FPL+6BjjpVSqcr223McPg1VjzSlZN8q0vd/5ef4Ho9FVtOu/tul212cDzolkPtkZrzfxLqd9qfiLT7+CeWHToNUjs4FjkKiY7vnY46jjA/GlVqKnYWFwcsRUcL2t/Vj1Gig5wcda4BYtYtfiTo/8Aa2pGd7iGcmGHKxIoHAA7n3NVKfK0rbmeHw/t+b3rWTfrZN/od/RRXm3i7UJ9X1LzbOeeC00m8igJjcqJ5WYbhx1CgfrSqVFBXKwmFeJqct7Lv+X3s9Jopqcxr9K4Dxvog0nQ7vXk1nUF1OIhoX88hGbPyp5f3fbpRUnyK9tBYWhGvUVNys3otL6v9P6seg0VXsHmk022e5GJmiUyD/axzXnfxH1K8v7e9j0+4lgs9JCNNLE5UyTNjC5HYA5PvilWqqlFyZWEwcsTX9knbz/D8z0yiorUk2cJPJ8teT9K8/8AGSaVYreTS67eHXRzaRJcMuHP3F2D5SPrTqT5FcWFw31ir7O/3K//AAy7s9FoqjopvG0OyOpkG8MK+cR0345rj/H1/NqKXek2E00C2EH2u6mhYqc/wx5HrnP4UVaipRcmGHwrr1/ZJ+r6dr/5HfUVS0di+i2bOSzGFSSTkniuf8UeG7e4tL3VL/Vr+F4Y2eJopzGkIAyPlHDc+tOpJwTdtiKVGE6vs5yt02udbRWH4Lu7y/8ABmmXWpsWuZYQzsRgtzwfyxWN48vbq7hl0fTZXi2W7XV5NG2GRF5VQRyCxH5A1NSooQ5jSnhJTxLoN7Npvpp1/wAjtaKxPBkjy+CdIkldpHa1QsznJJx3NYaWb+LvFWtQ6hd3KWGnSJBBDbytF8xQEsSuCevSqcnpZbhHCr2lSM5WUN3bztojt6K5fwNd3b2uo6df3LXMmm3r26Sv95kABXPvzTfGV9LcCPw/YSSRXN6jPJNGcGGJeSc9icYH1pOouTnQfU5fWPYX+flvf7jqqK53wG7yeC7AySPI20jdIxZjz3J61S8fW+qyaLeT22om0soLZnZIRiSRxngt2X6UpVLU+e3S4Rwt8S8O5W1tf52OvoqlozF9DsmYlmMCEknJPFZXi6+uhbw6RpTbb3UCU3g/6mMD53/Lge5q5y5EY06LqVfZJ/P03Z0VFcj8M3kfwZH500kzLcTKHlcs2A5A5Na2teHYtckX7be3kdui8RW8xiGf7xYc1MZuUFOK3szSph40sRKjOWkW1e3by/4JsUVxnw5u76e21W3up5Lmzs71oLOaXlmjXjk9/rWt4q1WSwsIrSzyb7UH+z24H8JPVz7Ac0c65OYqpg5wxP1dO77/AI/gt+xu0VyPw7ikttJvrWW5muTb3rxiSZyzHHua0PE9rYTW8b6zqk1lYg4ZI5THvbtlhz+FJVLwU+4p4ZQxDo817dUv0N6iuL+HlxJcf2r9muprrSI7gJYyTvuYgD5uTyRnpmt3xNqsulaSTZIJL24cQ2yertxn6DqfpT51ycwVcJKGI+rp3enlv37W69jXorhfhslzbza9a3l1Lcvb3oQtJIW52DOM9BnPFd1Tpz54KXcjFUPq9Z073tbX1VwoooqzmCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOX03SL2D4javqcsO2zuLaJIpNw+Yjrx1rb1mzm1DRbu0tpfJmmiZEk/uk1dorP2ceRw6O/46/qdM8ROdSNR7xSX3Ky/I8q1TRPEV54Cbw9Z+FYbZ1RQ8wuI/nwwPy+5x3qz4n06+1DxZpVhp0FleXEWmYntr9SYoxkfNwepIx+FemVha54TtNavor4XV3Y3kSbBcWkuxiuc7T2IzWc6V1pqerQzX94udKK956Xesklrd36dDF8OXN3Y6g3hjWdKsbEz27PC2nE7GXowIPIPNV9LsvEHhS2k0uDw9Dq9rHIzW1ws6I21jnDBvTNdJofhW00S5luhPc3t5MNrXN3Jvfb/AHR2A+lbdONJ2Tb1/QwrY6CnJQipRdr3uk2uq1ut+5S0n7Z/ZsZ1G3ht7g5LRQHKr+Nc34mOtXxu9NbwvDqdpIMQzm4VQuR/EDyCD6V2NFaThzx5bnFRxHsqvtOVP79PSzT/ABOEuvB+qxeDdGhtZYrjVtJZWQSn93IOjKT9K09OvteudRgSfwvBp8IIMs73CMQP9kLzmuoopezSd0zaWOnUi41Ip7666X1ezS373OEt7HxL4Y1zVv7K0yPU7PUZjPC5nEfkOeoYHkjntUHhCyutRtPGFjqDx/ari7kilaMYUM0YHHtzXoVZul6Jb6TdahcW7yM1/cG4lDkYDYAwPbio9lZ2vpZo3/tDmpz5opSfLqt3ZrfW2y6WucM9j4zn8DP4ZbR4IjHbmA3XnqyyoBgBV65I7nFa2seG9RNtoWp6bDFNqOlxLG9rMwCyqVwy56A+9drRT9inu+34EPM6nMnGKWrbtfXmVnfXr/wxzmj3mtXepqLzw5DpdsqndK06O7HHAUL/AFqt4B0W/wBFsNVj1KDyWuNSmnjG4HcjEYPFdZRVKGt7mEsW3CVOMUlK21+l+7fc8/g8H6jceBb7TpUW3vv7Qlu7XcwIz5hZCcZ4IrV02+8V381va3+iw6fAoxdTtcK+8YxhAOmfeuropKmls/6RpPHzqKXPFO7bW+je9tfzucBpun694SW40608Pw6zZea0ltMsyI4DHO1t3oT1rdurnxJb+E3msdLtP7W3Dbaq/wC7Az68c4roqKI0+VcqYqmNdWSnOCbvd76+uttetrGVrenS614VvNPkxHNdWxjOOisV/wAa48X+o3XgtPCkej3yal9nFlLI8WIUGNpcP0IxzxXotFEqfM733JoYz2UeVxvZ8y8n+q8jKudJmfwo2k2Vz9mlNsIEnC52HGM4rgtd8MeJbHTtCsYr6C6htbyIRCGyOYsH77fMcj1r1KilUpKpv5fg7lYXMKuHd0k1e+qW9rEVss6WsYu5FlmC/O6LtDH1A7V59qniBZvHel6jHperG3s4pklb7C/VgAMetejUVU4OVrPYzw2IjRlKUo3umt7bqz6FWUyX2lMbR2t5J4so0icoSO49favPdW8L+JtO8M29jHqdveRR3MZ2x2R3lt2d7Hdzzya9MopVKSnuVhsbPDP3Umr3s0nt52KunpdQ6bEuoTJPcKvzvGmwMfYZ4rzubXzqviQ3ev6RrBs7GXNnaRWLMpYf8tHPc+g7V6dRTnByas9gw+KjRlKUoXb87W720Kttc/2jpazwLJAZoyUEybWQ+615trvhHxJpfgW7sYtShv4WfzHijsyZZWZ8k7tx716nRU1aSqJp+g8Ljp4Wd4JWunZpPbbUo6NFew6TAmpzxz3AUZeOPYPyya5jxFrEV7p97pVz4cv5byZGiQC3DIxP3T5gOAM45PSu1oqpw51a5nRxCp1faSjd76aa/IxfCdhe6Z4Q0+y1A/6VDAFfndtPYZ74rldU8L+JbHw/rAh1a3ukui8skf2MmSTP8IO7sOBXolFTUpKcbPtY0pY6rTqyqJL3ndqy7367GR4Yt9QttAt49VnjmmCjBSLy9q44BGTyK47xLrUmqeIn07VNM1UaLaOCyW9oz/a3B7n+4OOO9ekUVU4OVrMKGLVOtKrKF29raW80UNG1GHU9OWe3tri1jB2CO4iMbDHse1cdf+FPEVjaa9cWmr28y6h5kskTWhaRhjAQHd2HAr0CilUpqorMmji5UJuVNKz6Oz2d+qOc8B2mo2fhGxi1SRSwhXy4/KKNEuPutzyfesuO5uPCfizWZbnTby5stSdJ4ZrWPzSGCBSpA6dOtdvRTcNFZ7DWMvUqTnG6nutut9PuOa8F6ffW1rf3uqw+Rc6hePceVkExqcBQSPYVTu/DfiGLVdU1Gx1i1AvFx5ctoXZEA4QHd9fxrsaKXso8qj2D67UVWVRJe90smrdtb9jlfh7Z6pZeGYk1WVSuP3UPkGNo+TkHJ5pPHmqCLQbzTI7K+uJ7q3YRm3tmkUE8YJHSuroodP8Ad+zT6WD62nivrE49b2WhieFdTS/0eKJba7t3t40RxcwNHk47Z61l3nhjXR4ovdZ03WLeP7REsSxzWxk8tAOg+YdTzXX0VUo8/wARMcVKnUlOkrKXR2fn1RxXwzsNWsdBddTcJEZpCkDQFHU7zkkk8g9RTfHetX0V1Bo1vZ6gLO5Tdd3lnAzsq/3Fx0JxyfQ129FR7K1NU09jX66pYt4mcE73dtte/wAjA8J39jPpwstM0++soLNFRRd25j3D2z16c1W1Xw7rVz4nGr6bq1vBsg8mOKe2MgTJySPmHJwK6iircLpX6fIxWKlCrKpTW/f3t990cX4CsNbs5tRbU54/Ia6kPl/ZyjM2R84JPQ+lbes6x/Zkyi60q5u7RlB823jEpDc8FOv4+9bNFTGnywUU9h1cUq1d1akVr0Wn5HF+DrKY+JdX1aCwl07TbwJ5cMy7GkcZy5X+HrVzXfDerX/ie11fTNVhtvssJjjhmgMignOW6jnHFdRRTVNKKj2LljqjrOrFJaW76Wt130OD8Cabrlp4g1yTUZkEL3ZLg2xTzm2j51OeB7c13lFFFOCpwUUZYrESxNX2kkltt5KwUUUVocoUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTXdY42d+FUEk+1OqG8/wCPC4/65N/KgcVdpHIf8Lc8H/8AQQl/8BZP8KT/AIW54P8A+ghN/wCAsn+FeWfDbwhYeMNQvbfUpbiNLeFXUwMASScc5Br0L/hSXh3/AJ/NR/7+p/8AE1wU6mIqR5opH2OLwGSYOs6NWU7r0/yNH/hbng//AKCE3/gLJ/hR/wALc8H/APQQm/8AAWT/AArO/wCFJeHf+fzUf+/qf/E0f8KS8O/8/mo/9/U/+Jq74nsjk9nkH89T8P8AI0f+FueD/wDoITf+Asn+FH/C3PB//QQm/wDAWT/Cs7/hSXh3/n81H/v6n/xNZ+v/AAh0LSvDuoX9vdX7S21u8qB5FIJCkjPy0nLEpXsi4UchnJQU53fp/kd94e8V6T4pink0Wdplt2Cybo2TBIyOorZryf4Ff8g/Wv8ArtF/6Ca9Yroozc6akzx80wsMJjJ0Kd7K2/omFFFFanmjXcRxs7dFBJxXnZ+Nnh4Ej7FqJwf+eaf/ABVeg3P/AB6Tf7jfyrwT4V+HdL8R69fwazai5iit96KWK4O4DPBFctac1KMYdT6DKcLhKtCtXxSbULbedzuf+F2+Hv8Any1H/v2n/wAVR/wu3w9/z5aj/wB+0/8Aiq2v+FXeD/8AoDp/39f/AOKo/wCFXeD/APoDp/39f/4qly4nujX2uRf8+5/ev8zF/wCF2+Hv+fLUf+/af/FUf8Lt8Pf8+Wo/9+0/+Kra/wCFXeD/APoDp/39f/4qj/hV3g//AKA6f9/X/wDiqOXE90Htci/59z+9f5mL/wALt8Pf8+Wo/wDftP8A4quk8JeNdP8AGK3TabDcRfZSocTqBndnGME+lc34x+HvhjS/Bmq31jpaxXEFszxuJHO0gdeTWV8Cf9Vrn+/D/JqmM6saqhNrU2rYTLa2XVMVhYyTi0tX5r17nrlFFFdp8oFFFFAEdxOttayzyAlYkLsB1wBmvOx8bvDpAIstRwf+maf/ABVd7q3/ACBb3/r3k/8AQTXh/wAI/DOk+JJtRTWrQXKwRRGMFmXaSTnoR6CuWtOopxhDqfQ5XhcHUwtbEYpNqFtvM7T/AIXb4e/58tR/79p/8VR/wu3w9/z5aj/37T/4qtr/AIVd4P8A+gOn/f1//iqP+FXeD/8AoDp/39f/AOKpcuJ7o09rkX/Puf3r/Mxf+F2+Hv8Any1H/v2n/wAVR/wu3w9/z5aj/wB+0/8Aiq2v+FXeD/8AoDp/39f/AOKo/wCFXeD/APoDp/39f/4qjlxPdB7XIv8An3P71/mYv/C7fD3/AD5aj/37T/4quo8J+MLDxha3E+mxXES28gjcTqASSM8YJrmvFXw78Lad4S1S8s9KWO4gtneNxI52sBweTWf8C/8AkC6v/wBfSf8AoApRnVjVUJtal4jC5dVy+picLGScWlq+9vU9UooorsPlwooooAyfFOqT6L4V1DUrRUaa2hLoJBlSfevHP+F0+Jz0ttOP0hf/AOKr1b4g/wDJPtZ/69mrkvgeit4b1Lcqn/TO4/2Frjrc8qqhGVtD6nLVhaWXVMTWpKbUktfOxy//AAunxR/z66f/AN+H/wDiqP8AhdPij/n10/8A78P/APFV7r5Mf/PNf++RR5Mf/PNf++RR7Cr/AM/Cf7Wy7/oDX3/8A8K/4XT4o/59dP8A+/D/APxVH/C6fFH/AD66f/34f/4qvdfJj/55r/3yKPJj/wCea/8AfIo9hV/5+B/a2Xf9Aa+//gHhP/C6fE4622nD6wv/APFV614I1y68R+EbTU75Y1nmLhhECF4cjgEn0rmvjRGi+BoyqqD9sj6D2atP4U/8k3076y/+jGpUueNZwlK+hpmH1WvlkcVRoqD57aa9GdjRRRXafKBRRRQAVw+ufFbRdA1u50u7tb55rZgrtGilTkA8ZYetdxXz74ntYb744S2l0nmQT38EciZxuUqgIrmxFSUIrl6s93JMHQxdaarp2jFvTyaO7/4Xb4e/58tR/wC/af8AxVH/AAu3w9/z5aj/AN+0/wDiq2v+FXeD/wDoDp/39f8A+Ko/4Vd4P/6A6f8Af1//AIqp5cT3R0e1yL/n3P71/mYv/C7fD3/PlqP/AH7T/wCKo/4Xb4e/58tR/wC/af8AxVbX/CrvB/8A0B0/7+v/APFUf8Ku8H/9AdP+/r//ABVHLie6D2uRf8+5/ev8zF/4Xb4e/wCfLUf+/af/ABVavhz4naP4m1qPTLG2vI5pFZg0qKF4GT0Y1L/wq7wf/wBAdP8Av6//AMVXmvgC2is/jRNbW6bIYZLlEXOdqgMAKhzrQlFSa1Z0Qw2VYrD1p4eMlKEW9X/wWe70UUV3HyQUUUUAZXiDxLpnhizjutZnaGKWTy0Kxs+WwT0A9Aa57/hbng//AKCE3/gLJ/hWR8cf+RW07/r+H/otqyPBfwt0XxH4Rs9Uvbm9SeffuWKRQowxHGVPpXHOrV9q4QsfUYXAZcsBHF4uUldtaW8/LyOu/wCFueD/APoITf8AgLJ/hR/wtzwf/wBBCb/wFk/wrO/4Ul4d/wCfzUf+/qf/ABNH/CkvDv8Az+aj/wB/U/8AiaL4nshezyD+ep+H+Ro/8Lc8H/8AQQm/8BZP8KP+FueD/wDoITf+Asn+FZ3/AApLw7/z+aj/AN/U/wDiaP8AhSXh3/n81H/v6n/xNF8T2QezyD+ep+H+Ro/8Ld8Hj/mIS/8AgLJ/hXZW88d1axXEJ3RyoHQ4xkEZFfP3xM8Faf4OaxXTZriUXKSF/PYHG3bjGAPWvdtB/wCRc03/AK9Iv/QBVUalSU3GfQxzTA4OjhqWIwjk1O+/l8kX6KKK6j54KKKKAEZlQZYhR6k0zz4f+eqf99CuI+MZI+HkuDj/AEmH/wBCrzbw38MNV8T6HFqlpqFrDFKzAJLu3DBI7D2rmqVpRnyRjc9/B5VQrYT61Xrcivba/n3PoDz4f+eqf99Cjz4f+eqf99CvFf8AhSGu/wDQWsfyf/Cj/hSGu/8AQWsfyf8AwqfbVf5PxNf7My3/AKDF/wCAv/M9q8+H/nqn/fQo8+H/AJ6p/wB9CvFf+FIa7/0FrH8n/wAKP+FIa7/0FrH8n/wo9tV/k/EP7My3/oMX/gL/AMz2rz4f+eqf99CpAQwypyD0Ir5x8XfD/UfB9hBd317b3CTS+Uqw7sg4J7j2r2v4ekn4eaIScn7KtVTrSnNwlGxhj8ro4bDRxFGrzqTttb9To6KKK6TwgooooAKKK80+LviXV/D50saNfPaed5nmbVU7sbcdQfWs6k1TjzM7MFhJ42vGhBpN9/JXPS6K8Kgvvitc28c9u2oSRSqHRxHFhlIyD0qTz/i36aj/AN+4v8Kw+tL+V/cey8gknZ4in/4F/wAA9xorw7z/AIt+mo/9+4v8KPP+LfpqP/fuL/Cj6z/df3C/sB/9BFP/AMC/4B7jRXh3n/Fv01H/AL9xf4Vk6h4y8e6NqCWmrajdW0xCv5ckceSpOM8D2NJ4tLVxZdPhyrUfLTrQb8nf9D6HopqHMak9SBTq7D5gKKKKACiiigAorB8ba3c+HfCF7qlisbTwBdokGV5YDkfjXlEXxf8AF06kwWNrKAcEx2rsB+TVhUrwpvlkevgsnxOOpOrStZO2rt/W57rRXh3/AAtnxn/0DIf/AACk/wAaP+Fs+M/+gZD/AOAUn+NZ/W6fmdv+reO7x+89xorw7/hbPjP/AKBkP/gFJ/jR/wALZ8Z/9AyH/wAApP8AGj63T8w/1bx3eP3nuNFeFy/F7xfCu6extolzjc9rIo/MmvVPA+uXXiLwjaanfrGs8xcMIgQvDkDgk+laU68Kj5YnFjcnxOCpKrVtZu2judBRRRW55AUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVDef8AHhcf9cm/lU1Q3n/Hhcf9cm/lSexUPiR418C/+Q3qv/Xsn/oVe114p8C/+Q3qv/Xsn/oVeoXsr3viWHTPOlhhS3M7+U5Quc7QMjnHescDHmpL5nucUTUMxm/8P5I26K4y9vbqPT7/AEo3Ekj/AG2O2SZjlgkmDjPU4BPNazRyXniE2IuZobW1tlPlwuU3MSRkkc8AV6Dw7Su3/wANp/mfLLFJuyXl87tfozdzWL4y/wCRI1n/AK8pf/QTWKbyWXSJLq4dnm0fUCgkzy6B9uD+BrZ8YNu8Daww72Mp/wDHDWWIpOnB38/6/E68vrqriIW7r87fmmcF8Cv+QfrX/XaL/wBBNesV5P8AAr/kH61/12i/9BNesVx4b+Cj3M//AORnV+X5IKKKK6DwyK5/49Jv9xv5V4v8D/8AkZtT/wCvX/2cV7Rc/wDHpN/uN/KvF/gf/wAjNqf/AF6/+ziuSt/Gh8z6TLf+RZi/SP5s9tpNwzjIz6UOdqMfQZrhkgW78Fz69KT/AGlh5luM/MmDwo9scYr0qVJT1btql958nWrezaSV3Zv7v+HO1N1ALxbUyL57IXCd9o4zUSanYyXhtI7yBrgZzEJAWGPaufsZnn1TWL1jmaK0jWMZ+7lNx/Wq13bw2fgWwubNFWdJIXSRR8xYuAee+cmtlh43UW9XZfN6nM8XLlcktEpP5Rt+Zb8c3cFz8P8AxCkEgdobZ0kHdTjOK4v4E/6rXP8Afh/k1bnjVjDp/iaFBhZ9JEz/AO8CRn8v5Vh/An/Va5/vw/yavKrw5cVC3b9D7TL6nPkOJ8ppfjE9cork/iJrN/omgQXGlz+RK9yELBQ2RtY45HtXm/8Awn/if/oKN/35T/CipiI05crR8TicypYap7Oad/K3+Z6td+LbOw8Vx6LejyfNiDxzs3yliT8p9Olb1fOmp6te6zdi51OczzBQm4qF4HTp9a67wh8Q5dL/ANE1x5rm2JAjlzlou3PqP5VlDFJytLY48Pm8JVHGpom9H/mepat/yBb3/r3k/wDQTXknwH/4+tX/AOuMP82r1e+uIrrw7dT20iyxSWzsrqcgjaa8m+BQJm1kL18iLH/j1VU/jw+Z+h5frlGK/wC3PzPUZfElrFMwMFybdX8troR/ugenXPrxnFWW1RP7bXTkTc3kmaR88IMgD865q3mjj+HctnK4+1BXhMRPz+YWIAx1zk1b06NxrmsROT9oFpCo7/wn+te1KhBKT7X+eyv+J8THEVG4r+a3y30/CxoR+JbSSZMQ3C28j7Eu2jxEx7YOc8njpVqy1E3V9eWskJiktXAHzZ3qRkMP1/KuZuZorvwBaWtoytcM8SLEp+YMHBPHbGCa1oA6+NXC9BYJ5mP7244/rSnRgk7Lv+FtfmKniajcb9eV/ffT5bh44/5EPWv+vOT+VcT8C/8AkC6v/wBfSf8AoArtvHH/ACIetf8AXnJ/KuJ+Bf8AyBdX/wCvpP8A0AV40/8AeI+h9vhv+RJX/wAUf0PVCQBknFN3p/eX865b4lMV8D3RUkHzI+Qf9sV4r5j/APPR/wDvo06uI9nLlsfCY3MvqtT2fLfS+/8AwD3LVfGdloviSHTL9fLhlg803W7IU5IAIA9utb1tdW97brPaTRzxN9142BB/Gvm4kscsST7nNbPh7xNqugXKjTpGeNiS1s2Sjk+3r7isY4t83vLQ4aOdP2j9ovdf3r/M9a+IP/JPtZ/69mrlPgb/AMi3qX/X5/7ItbnifUJ9T+FOqXV1Yy2Mj2rZhlPI/wA++Kwvgb/yLepf9fn/ALItaN3xEWux+j4WSlkVWS6yX6HWG91TUY728067SCG1kaNIGiDeYU6kt1GTxx6U6LVZdS1vTY7dzHD9mN1MAfvZyAv51FHbalpkd9YWti1wlzK8kNwHUIm/qGBOeDnoDSWtt/ZHiqxtm3GOXT/IV8cFlJJH5V7toWdrbafd1/4J8S5VLq999d/5la3l6DJtW1X+y7jW4p4xbQu2LTyx8yKcZ39ckc1ft9QMficWrSM0V5bCeIMc7SOCB+HNZzadqn9j3Ogi0+SV2VbzeNgjY55HXOOOn41aFp5nja1MZylhZFWPuxwB+QzRJU2mtOu3bS34/MlSqrlet7xvfvfW3y7aHP8Axq/5EWP/AK/Y/wCTVo/Cn/km+nfWX/0Y1Z3xq/5EWP8A6/Y/5NWj8Kf+Sb6d9Zf/AEY1eGv95fofd1P+RDD/AK+fozsaKKK6z5oKKKKACvAtc/5L4P8AsJ2/8kr32vAtc/5L4P8AsJ2/8krjxe0fU+n4c/i1v+vb/NHu11fWtjGHvbiK3RjgNK4UE/jSteW62ZuzMn2cLvMobK49c1BqT2NrbteX6xkRqQC4BPPYfWuXaynsvBUVtcIIBd3qZiB/1aPIDs/LivVp0ozS9Uv69D46tWlTb0urN/d/mdZPqNna26T3N1FDFJja0jBQfzofUbREgdp02XDBYmByHJ6YIrK8iK48aMtwiuLeyXylbkDcxBOPwFY24poetgD5LG/L24x90hgcCrjQjK2vb8TGeKlC913/AAVzt68N8Ff8lyu/+u93/wCzV7fCxeFGPVlBP5V4h4K/5Lld/wDXe7/9mrycR8cPU+zyV3w2Kf8Ac/zPcqKKK6j5wKKKKAPMvjj/AMitp3/X8P8A0W1b3wr/AOSbaX9JP/RjVg/HH/kVtO/6/h/6Lat74V/8k20v6Sf+jGrjj/vL9P8AI+nrf8iCn/jf/txtSa/BBdiKe2uoojIIxcvFiPcegznPJ46VPLqQj1iCwSPeXiaV33Y8tRwPzNZOo3cWtaqmmxSILW2kWS5mLjllOQg98gE1Kis3iXVVH3zZxiP6fP0/GvX9nFK7Wtv+G/zPi/azcmk9LpX/ADX6epY/4SS1+0BTBciAyeULry/3RbOOuc9eM4qzZaibq/vLSSExPbMAPmzvUjIYf57VzkU8X/Cv7e03r9q2rD5efmEm7GMeua1IQ6+M8DoLBRJj+9uOP61U6UEpWW1/wtr8zONeb5W+tvxvp8jzn48f63R/+uc3/sleqaD/AMi5pv8A16Rf+gCvK/jx/rdH/wCuc3/sleqaD/yLmm/9ekX/AKAK8en/AB5/I+1x3/IowvrL8y/SO6xxs7nCqCSfQV534v8AH2qaD4kmsLOG2aKNEIMiknlQexrBm+KWtzQPE1vZhXUqcI3Qj61csRCLaPh6maYenJwd7ryPWrDUbTVLRbrT7hLiFujocirNfPeh+INR8PXAk02cqucvE3KSfUV7F4Z8Z6f4igVVcQXiqDJAx6HOOD3/APr0qWIjPR6MWDzKnifdlpL+tjF+Mf8AyTyX/r5h/wDQqsfCX/knNl/10l/9DNV/jH/yTyX/AK+Yf/QqsfCX/knNl/10l/8AQzU/8xPyPuH/AMiFf9fP/bTtKM1zmqRz6ddWH2S+uZbqa6AaN5CweM53fL0AA7gVJLKP7dv75ssNPtdqLzjcQWJ/LivT9jdXT/rY+Tde0uW2un4m/RnNcozXNvoFnrIvblpneOWVGkJRlcjK7eg6+narumubfxZqNoCfLmiS6Vc8KT8rfyFDoWTae36Wv+ZKxKbirb2+53t+RyHxx/5FnTv+vz/2Rq6n4ef8k70T/r1WuW+OP/Is6d/1+f8AsjV1Pw8/5J3on/XqteZH/eJeh9fX/wCRHR/xv9TpKKqXuq2GnFBqF7BbF87RLIF3Y64zVb/hJ9C/6DFj/wCBC/4108yW7Pl3UhF2bRqUV51efEn+zPFt1AxS+0r5djwEFl+UdD0Izmu603VLLV7RbjT7hJ4yBnaclfYjsfapjUjNtIyo4qlWk4weqLdeQfHb72i/Sb/2WvX68g+O33tF+k3/ALLWWK/gs+n4d/5GdP5/kz0vwz/yKek/9eUP/oArUrL8M/8AIp6T/wBeUP8A6AKg1C81bT5VuZGtGtTOsfkhW37ScA7s4z7YrqpQc0kjxMXNQqTbXVm3RWTJqEj+IhaROUht7czT8D5snCj9CaoR65qMljHq6rANOeTHlFT5gTdjfuzj3xj8a2VCTX9ddjkliIRbXb9LXfyudLXgvxk/5KFD/wBecX/obV7FY6hOfEl7YXD74/LWe3OMYQ8EfmP1rx34yf8AJQof+vOL/wBDavPx8HCnZ+R9XwpNTx7a7SR7xH/qk/3RTqbH/qk/3RTq6T517hRRTJ5RBbyTMCVjQsQO4AzQLYfRWJofi3SNfhU2dyqTHrBKQrj8O/XqK26UZKSuiIVIVI80HdHIfFT/AJJrqn0j/wDRi1g/A3/kV9S/6/f/AGmtb3xU/wCSa6p9I/8A0YtYHwOOPC+pf9fp/wDRa1yS/wB5XofVUf8AkQVf8a/9tPTqKx28RJFeww3VjdW8dxJ5UU8ijazduM5GfcCrSamsmtyafGmfJhEkkm77pJ4XH05r0HSmuh8mq0HpfyL1FZOn+IrPU9WnsLVZS0KbzKy4RhnHHrz3qxYag13dXlvLD5UlrIFxuzuUjIYcf5xRKlOO6CNanP4XfWxxnxp/5ENP+v2P+TVofCn/AJJvp3+9L/6Mas/40/8AIhp/1+x/yatD4U/8k307/el/9GNXAv8AeX6H1M/+RDH/AK+fozsaKKK6z5sKKKKACiiigAooooAKKKKACiiigAooooAKhvP+PC4/65N/KpqhvP8AjwuP+uTfypPYqHxI8a+Bf/Ib1X/r2T/0KvVNS0+8/tSHUtKMJuFQwyJOxCsmc9QDgg15X8C/+Q3qv/Xsn/oVe11hgZuFNNeZ7vFEFPMZp/3fyRymqaRdW2hTXj7Zr4XSXkojzj5SPlH0UYrQurPUDqceqaM0DGaARyxXBKgjqCCAeea26K9D6xK2q7/pp+B8t9Vhe/p96bd/x1OVvdOWy0mLTTmS51O7DTMo4yW3OfoMVf8AGA2+BtYUdrGUf+OGtusXxl/yJGs/9eUv/oJrGtUc4WfmdmBoxp4mDXeK/H/NnA/Ar/kH61/12i/9BNesV5P8Cv8AkH61/wBdov8A0E16xXLhv4KPaz//AJGdX5fkgoooroPDIrn/AI9Jv9xv5V4v8D/+Rm1P/r1/9nFe0XP/AB6Tf7jfyrw/4OajZad4i1GTULuG1RrbarTSBATvHHNclbSrD5n0uVxcstxaSu7R/NnupGRg1z48LlYXsY71l0uRyzWvljPJyQHznGfarf8AwlGgf9BrT/8AwJT/ABo/4SjQP+g1p/8A4Ep/jXbCs4fCz5yeCqVPig/uY2HS5bfxJPdR7TZ3NuEkTPRl4H4YzS2/h2ztlijWSd7eF98UDvlEPbHGeO3NL/wlGgf9BrT/APwJT/Gj/hKNA/6DWn/+BKf403iH/MSsBJf8u39z67nP+MrGdPCviq+ulUebaGOEA5IRR3+pJrmPgT/qtc/34f5NXU+OfEOjXXgTWILbVbKWWS1dUjS4VmY46AA81y3wJ/1Wuf78P8mrhqz58TB+X6H1GEoyo5FiFJNXknr6xOs+JWm3uqeHbeHTrWS5kW5VikYyQNrc/rXmH/CI+Iv+gLef9+69/oqqmHjUlzNnwWKyyniantJSaPnC90+70248jULaS2l27tkgwcetdP4L8Dy+IZPtd/vh09CRxw0p9B7e9eh6j4Nt9W8XR6vqDiSCKJVW3x1YEnJPcc9K6QAKoCgADoBWUML715bHFh8nSquVT4U9F39TNuLG303wvcWdlGIoIbV1RR2+U15Z8B/+PrV/+uMP82r1vVv+QLe/9e8n/oJrxr4KanY6bcaodQvILUPFEFM0gTdgt0zVVLKtD5n6PlsG8pxUYL+T8z2c6bYm8+1m0g+0/wDPbyxu9OvWojpUY10apG5WQwmKRccOM5H5VB/wlGgf9BrT/wDwJT/Gj/hKNA/6DWn/APgSn+NdqrW+15HzrwdR/wDLt99upcTTbKO8N3HaQLcNnMojAY/j1qOy077LfXl08xlkumB5GNigYCj9ar/8JRoH/Qa0/wD8CU/xo/4SjQP+g1p//gSn+NHttLXH9TqXvyPvsyt44/5EPWv+vOT+VcT8C/8AkC6v/wBfSf8AoArpfGXiLRbnwVq8Nvq1lLLJaSKiJcKSxx0AzXNfAv8A5Aur/wDX0n/oArjk08RG3Y+joQnDJa6kre9H9D0nU9MtNYsXs9Qi82ByCybiM4ORyKw/+FdeGP8AoG/+Rn/xrp6K6HCMndo+SnQpVHecU35o8r8RfDuWbxJDa+HbMW9o1vueaRmKB8njPJzjHFdf4Z8Eab4c2zgfab0Ag3DjpnsB0FdLRURowjLmsc9LAUKVR1Etfy9DnPiD/wAk+1n/AK9mrlPgb/yLepf9fn/si11fxB/5J9rP/Xs1eYfDLx1o/hTR7y21Y3AkmuPMTyo9wxtA9fasKklGvFvsfcYGhVr5NVhSjd860XyPcqa8UcjIzorMhypIztPtXCf8Lk8K+t7/AOA//wBej/hcnhX1vf8AwH/+vW/t6f8AMeR/ZGP/AOfMvuO9piRRxu7oiq0hy5AwW+tcL/wuTwr63v8A4D//AF6P+FyeFfW9/wDAf/69Ht6f8wf2Rj/+fMvuIvjV/wAiLH/1+x/yatH4U/8AJN9O+sv/AKMauG+I/wAQdE8UeGEsNLNwZluUkPmxbRgA55z713Pwp/5Jvp31l/8ARjVhCUZYhuL6HsYqhVw+SQhVi4v2mz9GdjRRRXafKBRRRQAV4Frn/JfB/wBhO3/kle+14B4hljg+OzSzOscceo27O7HAUBUySa48X8MfU+o4b1q1v8D/ADR7hqWjwapJA1xJKvkNvQIRjd64INV9W0ia80Oa1huZJJ9wkieYjhlII6AcZFO/4SjQP+g1p/8A4Ep/jR/wlGgf9BrT/wDwJT/Gu6NdxtZ7HzMsDOV7weu+jJJtJju54LyR5be8jj2F4XA4PVTkEEZqvfaO39nwWNgg8p7hXuHduWXOWJ9Sak/4SjQP+g1p/wD4Ep/jR/wlGgf9BrT/APwJT/GmsQ1bXYTwM2n7j18jVAAAA6DpXhvgr/kuV3/13u//AGavXf8AhKNA/wCg1p//AIEp/jXkHgWRJvjdcSROrxvNdMrKchgd3INcFeSc4W7n02UUpww2K5ote4/1PdKKKK7D5gKKKKAPMvjj/wAitp3/AF/D/wBFtW98K/8Akm2l/ST/ANGNWD8cf+RW07/r+H/otq3vhX/yTbS/pJ/6MauOP+8v0/yPp63/ACIKf+N/+3G+2g6Q8hd9MtGYnJYwrkn8qc+lqdZh1CKTy3jiMLoF4deoHtg1eor0PaT7nynsodisum2KXhu1tIBcnkzCMbj+PWorPTvst/eXbzGWS6YdRgIoGAoq9RS55Wtcfs43vbzPHPjx/rdH/wCuc3/sleqaD/yLmm/9ekX/AKAK8r+PH+t0f/rnN/7JXqmg/wDIuab/ANekX/oArhp/x5/I+nx3/IowvrL8zzPx94e1jUPGFxcWOm3FxCyRgSRpkHCjNc0/hPxAiM76PdqqjJJj6CvoCmTRCe3kiYkCRSpI9xiiWFjJt3Pga2UU6s5VOZ3evQ+b7a3mvLhILSJppZDhEQZLV6v4P+HcWmSQ6jrG2a7UBkhx8sLZ657np9K6Pw74W07w1Z+TYoXdjl5pMF2/GtmlRwyjrPcWCyqNG1SrrL8EcH8Y/wDknkv/AF8w/wDoVWPhN/yTiyx/z0l/9DNV/jH/AMk8l/6+Yf8A0KrHwl/5JzZf9dJf/QzT/wCYn5H3z/5EK/6+f+2l+207XodXmv549PuJn+RXaZx5cefugbf171Y2BPEt3ZzIfK1C2DBhnkjII/Kt2ivTdZvddLHyaw6Wz63/AMznIdF1R7eDTbuS2/s63cEMhJkkRTlVIIwOgyc9qm02M3HivUrwKRHDGlqhI6kfM38wK3aKTryad+v9MFh4pq3T9L2XyueZfHH/AJFnTv8Ar8/9kaup+Hn/ACTvRP8Ar1WuW+OP/Is6d/1+f+yNXU/Dz/kneif9eq15sf8AeJeh9bX/AORHR/xv9Tkvi/8A8fmk/wC5L/Na85r6H1PQdL1lo21SyiuTECEMg+7nr/KqP/CD+Gv+gNbf98moq4eU5uSZ+dYzKqtevKpGSs/8jwatnwvPrUWtRp4dZ/tLnBUcoR/tDpjrXZXXw0a+8XTmONbHRwBt8ojcx2joO3Pc13ukaLYaHZC202BYk/iP8TnGMk9zWdPDTctdLHLhcqrOpeT5Unv1fp/mT2BuzYQnUVjW62jzRESVz7V5R8dvvaL9Jv8A2WvX68g+O33tF+k3/stdOJ/gs/SOHVbMqS9f/SWel+Gf+RT0n/ryh/8AQBWRdXU974gDahpeofZLOTNuscJYSP8A3zz0Hatfwz/yKek/9eUP/oArUrsoVFBXt0PDxtN1Ksle2r/M52OPHjK/jcEfa7FChI9CQR+oqla218vh+Lw9JYzCVGEb3G391sDZ3BvXHb1rqpLaGWeOaSMNJFnYx6rnrUtb/WLJadvvWxxPDXbbff7na/4nPWw83x3MYzlbWxWJj7ls/wAq8m+Mn/JQof8Arzi/9DavdYbWCCSWSGNUeZt0jDqxxjJrwr4yf8lCh/684v8A0Nq87MJqdNW6WPreE6fJj3fqpP8AL9D3iP8A1Sf7oqjr9zLZ+HdQubZ9ksNu7o2M4IUkGr0f+qT/AHRUOoWSajptxZSsypcRtGzL1AIxxWrvbQ+cqJuLUdzxMePvE2P+Qo//AH7T/CmyeO/EksTRyam5RwVYeWnIP4V3H/CpNK/6CF5/45/hUV18KtLt7OaZb+7JjjZwDt5wM+lef7Kv3/E+VeDzG2sn/wCBHlisUZWQlWUgqwOCD6iu80D4oXtm3la6hvIcfLJGAJF/oawfD3g3VfERR4I/ItW/5eZB8vTPA6n8K9U0DwLpGgt5qRtc3BGPNnwcfQdqmhTq3vHREZdhsW5c9N8q8+vy6md8SbkXnwqv7hY5I1kSNgsq7WH7xeorH+B3/Ir6l/1+/wDtNa3vip/yTXVPpH/6MWsH4G/8ivqX/X7/AO01rof+8r0P1ah/yIKl/wCdf+2nUSXEPiHWo0EoSy0+bcSWA86UHgD2BqfRio8Ta6G+/wCbGfw8sVsCztg24W8Wc5zsFQrpkKaw+ooWWWSIROo6MAcg/WvW9rFxcVorW/FM+M9jPmU3q73+VmvwuY8E8J+IVwRKhBsUAIYcne3FWrUMfGd6U/1YtYw3+9uP9K1FtLZWDLbxBhyCEGRUVlpyWU91MJHkkupPMct24wAPak6sWn6WBUJJ/wDb1zifjT/yIaf9fsf8mrQ+FP8AyTfTv96X/wBGNWf8af8AkQ0/6/Y/5NWh8Kf+Sb6d/vS/+jGry1/vL9D7Cf8AyIY/9fP0Z2NFFFdZ82FFFFABRRRQAUUUUAFFFFABRRWbrOv6foEcMmqStEkz7FYIWAOM84pNpK7JlOMFzSdkaVFVLDVLHVIBLp91FOhUN8jcgHpkdR+NW6E09UNSUldBUN5/x4XH/XJv5VNUN5/x4XH/AFyb+VD2Lh8SPkyF5U/1DSKcc7Cf6VL597/z0ufzavSfgYAdb1TIB/0ZOv8AvV7TsT+6v5V5NHC+0hzXsfpOZ8QLA4mVB0ua1tb23Xoz5N8+9/56XP5tR597/wA9Ln82r6y2J/dX8qNif3V/KtvqT/mPO/1tj/z4/wDJv/tT5N8+9/56XP5tSNNdlSHkuCpHO4tivrPYn91fyrG8Yoo8E6zhR/x5S9v9k1MsG0r8xpS4qjUqRh7Ddpb/APAOC+BX/IP1r/rtF/6Ca9Yryf4Ff8g/Wv8ArtF/6Ca9Yrrw38FHzuf/APIzq/L8kFFFFdB4ZFc/8ek3+438q+VLHSb/AFeeSLTLKa8kQbmWFNxUZ619V3P/AB6Tf7jfyrxf4H/8jNqf/Xr/AOziuHEwU5wi/M+vyDFSwmExNeKu48v6nFf8IV4l/wChfv8A/wABzR/whXiX/oX7/wD8BzX1FRS+ox7mn+t2I/59r8T5d/4QrxL/ANC/f/8AgOaP+EK8S/8AQv3/AP4DmvqKij6jHuH+t2I/59r8T5ZufCmvWVtJc3ei3kEMS7nkeEhVHqTXpXwJ/wBVrn+/D/Jq7r4gf8k81v8A683/AJVwvwJ/1Wuf78P8mqI0VSrxSOrEZnPMcmrVJxSs0tPVHrlFFFekfBBRRRQBU1b/AJAt7/17yf8AoJr5Z0vRtR1dSul2E94Y1BcQxltuemfyr6m1b/kC3v8A17yf+gmvJPgP/wAfWr/9cYf5tXDiIKpUhF+Z9hkWLlg8DiK8Vdxcf8jg/wDhCvEv/Qv3/wD4Dmj/AIQrxL/0L9//AOA5r6emnit1DTypEpOAXYAE+nNJNdQ27xJK+1pn2IMZ3Gj6hF7NlPi+ut6cfvZ8xf8ACFeJf+hfv/8AwHNH/CFeJf8AoX7/AP8AAc19OPeWsdwIJLmFZm6RtIAx/DrQl3DJdS2yPmaIAuuCMA9D+lH9nruw/wBcK97ezj97Pl+48J6/aW8lxc6JewwxqWeR4SAoHcmvUvgX/wAgXV/+vpP/AEAV23jj/kQ9a/685P5VxPwL/wCQLq//AF9J/wCgCohRVKvFJnRiszqZjlFWc4pWaWnqj1SiiivRPhQooooA5z4g/wDJPtZ/69mrw/wr4A1TxfZT3WmzWsaQSeWwnZgScA8YB9a9w+IP/JPtZ/69mrlPgb/yLepf9fn/ALItcNaCqV4xl2Pr8sxdXB5TVrUt1Jfocv8A8KT8R/8AP3p3/fx//iaP+FJ+I/8An707/v4//wATXuck8UToksqIznCKzAFj7etRte26XyWbSD7Q6F1TB+6O9afUqb2TOR8UY9byj9x4h/wpPxH/AM/enf8Afx//AImj/hSfiP8A5+9O/wC/j/8AxNe5iaMzGISIZAMlNwyB64qO1vbe9837NJv8lzG4wRtYduaPqVO17MP9aMfe14/cfO/if4dav4U0ldQ1Ge0kiaURAQuxbJz6gelevfCn/km+nfWX/wBGNWd8av8AkRY/+v2P+TVo/Cn/AJJvp31l/wDRjVlSpxp4hxj2O3MMZVxuTRq1t+e34M7Giiiu8+OCisXW/FNh4fvrSDUy0aXQbbKBkKQR1H49fatiORJY1kiYOjDKspyCPWkpJuyIjUhKTinqtx1fN3xDhkufifqsFvG0ssk6IiKMliY1wAK+ka8C1z/kvg/7Cdv/ACSuPGK8YrzPruGJuniKs10g3+KOa/4QrxL/ANC/f/8AgOaP+EK8S/8AQv3/AP4DmvqKip+ox7nR/rdiP+fa/E+Xf+EK8S/9C/f/APgOaP8AhCvEv/Qv3/8A4DmvqKij6jHuH+t2I/59r8T5d/4QrxL/ANC/f/8AgOa3/hRDJb/Ey3hmRo5I45ldGGCpCnINfQdeG+Cv+S5Xf/Xe7/8AZqzlh1SnBp9Tto5zVzLCYmM4pcsHt8z3KiiivTPz8KKgnvbe2uLeCeUJJcsUiU/xkDJH5Cp6BXTPMvjj/wAitp3/AF/D/wBFtXi0ctyqARPOF7BC2P0r2n44/wDIrad/1/D/ANFtW58LFU/DfTCVB4k7f9NGrzalL2tdxvbQ++wGYrL8nhVcOa8mrXt38n2Pnzz73/npc/m1Hn3v/PS5/Nq+stif3V/KmsYVkVGMYd/uqcZb6U/qL/m/Az/1tj/z4/8AJv8A7U+T/Pvf+elz+bUefe/89Ln82r6y2J/dX8qajQSlhGY3KHa20g7T6Gj6i/5vwD/W2P8Az4/8m/8AtT5KuHmdD57SNgHHmE/1r6q0H/kXNN/69Iv/AEAV5X8d1Al0fAA/dzdB/uV6poP/ACLmm/8AXpF/6AKrDQ9nUlH0MM+xf1zA4evy8t3LT8C/RRRXefGhRRRQBwfxj/5J5L/18w/+hV5DpPjvxFoenJYaXf8Ak20ZJVPKRsZOTyR61698Y/8Aknkv/XzD/wChVlfDjwV4d1nwRa3uqaVDc3LvIGkctk4cgdD6V51aE517QdnY+4yzFYbC5Rz4mHPHnatZPW3mcH/wtLxh/wBBX/yAn+FH/C0vGH/QV/8AICf4V7L/AMK18H/9AK3/ADb/ABo/4Vr4P/6AVv8Am3+NH1ev/P8Aiyv7ayb/AKBv/JY/5njX/C0vGH/QV/8AICf4Uf8AC0vGH/QV/wDICf4V7L/wrXwf/wBAK3/Nv8aP+Fa+D/8AoBW/5t/jR9Xr/wA/4sP7ayb/AKBv/JY/5ng+ueMNc8R20dvrN79oiifei+Wq4OMZ4HvXvnw8/wCSd6J/16rXn3xY8KaH4f0Gyn0bTorSWS52OyE8rtJxyfavQfh5/wAk70T/AK9Vp4eMo1mpO7sZZziKGIyylUw8OWPNtZLo+x0lFFFegfFBRRRQAV5B8dfvaL/22/8AZa9fryD46/e0X/tt/wCy1zYr+Cz3uHf+RnT+f5Mp6b8aZdO0q0shoaSC2hSLf9qI3bVAzjb7VZ/4XtN/0L6f+BZ/+IqbSfgzpuo6NZXsmq3aNcQJKyqi4BZQcdPern/CjdL/AOgvef8AfCf4VzJYq2j/ACPeqVOHOd80db6/F/mZv/C9pv8AoX0/8Cz/APEUf8L2m/6F9P8AwLP/AMRWl/wo3S/+gvef98J/hR/wo3S/+gvef98J/hTti+/5Ee04b/l/9L/zM3/he03/AEL6f+BZ/wDiK4bxd4obxd4ii1J7QWhEaQ+WJN/Ric5wPWvS/wDhRul/9Be8/wC+E/wrzrxx4Yg8JeKItNtbiS4QwpLvkAByWIxx9KxrKvy/vNvkeplc8neItgladn/Nt8z6Tj/1Sf7op1Nj/wBUn+6KdXrn5m9woIBGCMg0UUCEVVRQqKFUcAAYApaKKAOQ+Kn/ACTXVPpH/wCjFrzX4dfEHTvB+j3dpf211M81x5qmEKQBtA7kelelfFT/AJJrqn0j/wDRi15D4P8Ah5e+MtOnu7O/t7ZYJvKKyoxJO0HPH1rzq7mq65N7H3GTwws8pmsW7Q5/0jbY9D/4XfoP/QP1D/vlP/iqP+F36D/0D9Q/75T/AOKrnf8AhRmrf9Bmy/79PR/wozVv+gzZf9+no58V2K+rcOf8/Pxl/kdF/wALv0H/AKB+of8AfKf/ABVH/C79B/6B+of98p/8VXO/8KM1b/oM2X/fp6P+FGat/wBBmy/79PRz4rsH1bhz/n5+Mv8AIi+IHxI0zxZ4bXTrG1uopRcJLumCgYAPoT616B8Kf+Sb6d/vS/8Aoxq8o8WfDS+8I6MNRu9Qt7lDKsWyJGByc88/SvV/hT/yTfTv96X/ANGNRQc3XftN7EZtDCQymKwbvDn897PudjRRRXonxAUUUUAFFFFABTJ2KW8jLwVUkflT6ZOpe3kVeSykD8qBPY8T/wCFi+J/+f8AT/vwn+FJ/wALF8T/APP+n/fhP8KP+FdeJ/8AoHr/AN/0/wAaP+FdeJ/+gev/AH/T/GvJ/f8An+J8X/wof3/xD/hYvif/AJ/0/wC/Cf4Vnax4n1bXoY4tUuVmSJtyARquDjHatH/hXXif/oHr/wB/0/xrO1jwxq2gwxy6rbCFJW2oRIrZOM9qmXtbe9exnV+u8j9pzW872M6C4ntZN9rNJC+Qd0bFT+lddpPxO1nT0Ed6sd/GMAeZ8rgD3HU+5zXIQW811JstoZJnyBtjUsf0rr9J+GWs6ggkvGjsIzgjzPmYg+w6H2OKVP2l/cJwn1rm/wBnv+n+R2WkfEzRb9VW+LWEuBkSDKZPYMP5nFdM13b32kTT2c8c8LRvtkjYMp4PcVzOk/DLRbBVa+D38uBkyHCZHcKP5HNdM1pb2OkTW9nBHBCsb7Y41CqOD2FenT9pb3z7DBfWrr6xb5b/AOR5B8C/+Q3qv/Xsn/oVe114p8C/+Q3qv/Xsn/oVe11nhP4SPreJP+RlP0X5IKKKK6j50KxfGX/Ikaz/ANeUv/oJrarF8Zf8iRrP/XlL/wCgmpn8LOjC/wC8Q9V+ZwPwK/5B+tf9dov/AEE16xXk/wACv+QfrX/XaL/0E16xWOG/go9TP/8AkZ1fl+SCiiiug8Miuf8Aj0m/3G/lXi/wP/5GbU/+vX/2cV7Rc/8AHpN/uN/KvF/gf/yM2p/9ev8A7OK5K38aHzPpMt/5FmL9I/mz22sDxJ4hl0hdllEk06r5kgfOETOMnHcnp+Nb9cd4g0PVf7P1KWC6juTcuD5YtSZNoIwoIboPpXqYaNOVRc+x8hi5VI0n7Nam6mrl9bks1RfKgt/Nmk5+UnoPyyapxeJJ2jgvLjT/ACdOnkCJMZcuMnCkrjgE+9VtOjkOra5aTurXUlvEQVG3cNmOB25qrLN/anhOy0uBHW8MkSPCykGPawLE+3FdCowva2mmvqrtnLKvVs2nr72nmrJIPGWoSz+D/FNncIitb2pMZX+NGHB/MEVy3wJ/1Wuf78P8mrc8boWsvErpykWjiN8f3ssR+lYfwJ/1Wuf78P8AJq8fERSxULdv0PtsulKWQ4m/Sa/OJ65RRRW588FFFFAFTVv+QLe/9e8n/oJryT4D/wDH1q//AFxh/m1et6t/yBb3/r3k/wDQTXknwH/4+tX/AOuMP82rkqfx4fM+kwH/ACKcX/25+Z1XijU7S/kvUuLmFEsWCQxM43PJkbmx7DgfjW2t2l74ztfKkWSCOyeRCrZDEsoz+Wa1tQ0y31Kze3nXar4yygA9c1VudOkXX7PULZdyojQyqMDCnBz+BFeyqtOUFFaWT/L9T4t0akZubd7tfg/0RzlrDFe+BdRv7lVe7ZpZPOP31ZemD1GMCr0dzIviHQJD9+7smWX3wFYfqT+daL+GbVpJFWadLSZ98torDy3buemeeOAcU8adJL4kju3jCW9pb+VByOSx5PtgACrdaEr/AD+V1ZL7zL6vUikuunztK7f3fmV/HH/Ih61/15yfyrifgX/yBdX/AOvpP/QBXbeOP+RD1r/rzk/lXE/Av/kC6v8A9fSf+gCvDn/vEfQ+5w3/ACJK/wDij+h6pRRRXWfNhRRRQBznxB/5J9rP/Xs1cn8D2C+GdTZuALvJ/wC+FrrPiD/yT7Wf+vZq5P4G/wDIt6l/1+f+yLXJL/eI+h9Jh/8AkR1v8a/QvXurafqGpWmpy3sIK3yJFE0gzHGpOWI7ZPP0xXUaftufEmoXBwWjRIkOf4fvf1q5e6VbXxgMq7fIlEq7QBkjsfaoEtbmDxJJPGoNpcQAOQQNjqTzj3FezOrCcLR0sn+aPio0Z06nNLW7W3z/AOAUbKxgsfGcq26keZab3ZmLFiXPUmrFk3k+LtQt0ACSwxztj+9kr/ICnR6AyaqL86peNJjaVOzBXOdv3elSaZa3A1K/vbyIRtK4SIZBPlqPb1JNTKaabvfS34lRpyi0kre9f5WOR+NX/Iix/wDX7H/Jq0fhT/yTfTvrL/6Mas741f8AIix/9fsf8mrR+FP/ACTfTvrL/wCjGryF/vL9D7Sp/wAiGH/Xz9GSfEPXNQ0HR7afS5hDJJPsYlA2RtJ7/SvPP+Fi+J/+f9P+/Cf4V6H8Q9D1DXdHtoNLhE0kc+9gXC4G0jv9a88/4V14n/6B6/8Af9P8ayr+15/dvY/Ncx+ufWH7Lmtptexl6z4h1LX2hbVZxMYQQmEC4zjPT6Cr/hnxjqHh25VRI09kWHmQOc8f7Poaoaz4e1PQGhXVYBCZgSmHDZxjPT6ir/hnwdqHiK5VljaCyDDzJ3GOPRfU1zL2nPpueTT+tfWPdvz/AI/M9g0PxHp3iG1EunzgvjLwsR5kfOOR2rxfXP8Akvg/7Cdv/JK9o0Lw3p3h62EWnwAPjDzsB5knOeT3rxfXP+S+D/sJ2/8AJK68Rzckebe5+vcLe05qvtd/Zvb1R76SACTwB1rk0v7m+8UWF3FPKllK8kUcQJCyKqk7yO+T0+ldVJGs0TRyDKuCrDOMg1hyeF7db6wktWljhtSxKG5kPGMADnivXoSpxvzdn+R8liI1JW5Nrr80SaVM97q+qz5/1Mgt485wAFyTj6t+lVdBN7H4n1e3vb2S62pE6g8Km7dkKOw4qzpEUllrGrW5X/Wyi4iJJwwK4P6r+tFjp2pW+v3d/MtrsulRWVJGJTbnp8vPWtG0uZd0rfh/wTFJvlet1J3/AB/4Amh3JXWNW04klbeYPHk9FcA4/MmvKfBX/Jcrv/rvd/8As1eraHbFtY1bUSCFuJgkeR1VABn8815T4K/5Lld/9d7v/wBmrzsdb20LeX32Vz6vh3m+p4vm/ll913Y9q1C5az0y6uUAZoYXkAPQkKT/AEry3/hberf9A+z/APH/APGvTtYjeXQ7+ONS7vbSKqqMkkqcCvBv+Eb1z/oD33/gO3+Fc+InOLXKfD5pWxFOUfYt/I0vEnjW+8Sw28dxBDb/AGeQyK0JYHOMdzXQ+DviM8DQabrpBhA2Jdk8r6bvb3rhbvS7/T0Vr+yuLZXOFMsZUE+gzVzw94bvvEeoJb2iMsWf3lwV+WMf1PtXJGpU57rc8WjicUsRzK7k+nc6/wCNU8V14P0ua2kWWKS8DI6HIYeW3INdF8K/+Sa6Z9JP/RjVxvxP0SPw/wDDzSLCOV5tl7lnYnklHzgdh7V2Xwr/AOSa6Z9JP/RjV1Qu8Q79v8j9cm5Ph6k5Kz5/0ZHrN3/aV1FewXRFtbXkUMaq+A7bvmb3Hb8K2YgLvxjM7ciztkWP6uSSfyUUzUPCOlXkSLDZWsDLKsjMsI+YA5I/GnrbSWXipXggP2W5thGxReI2QnGfQEN+le46lOULQ3Sf6f8ABPh/Z1I1HKa0bj+b/LQrQ2bWfjVGa6nnNxDI7CRztXkYAHQAVPa7bTxldwRjC3Vus7Y7MCVJ/Hilk0nU31uO/wD7QtwIwUWP7MfuE5xnd1460+xikuPEl9ezQvGsSLbws4xvAyWI9sn9KlyTV73939dClCUZWSteV192vfzPNPjx/rdH/wCuc3/sleqaD/yLmm/9ekX/AKAK8r+PH+t0f/rnN/7JXqmg/wDIuab/ANekX/oArx6f8efyPtcd/wAijC+svzL9FFFdZ82FFFFAHB/GP/knkv8A18w/+hVY+Ev/ACTmy/66S/8AoZqv8Y/+SeS/9fMP/oVWPhL/AMk5sv8ArpL/AOhmuT/mJ+R9G/8AkQr/AK+f+2mr4j8RS6RiOyiSaVQHl35wiE4HTuT0+hqzc6hOfEVlYW+Ahjaec4z8vQD86wvEGhaqun38kFzHcm5lVigtSZMAjAyG6D6VfjZ7fxpbrdkM89h5YcJtBZWyQPwr2/Z0vZpx1dn+S/I+JdSt7VqV0rx/Nr8Rza7qB0671NIIBZ28jAIcl5EU4LZzgeuMGrVvqMra+IXlD2t1bCa3GMEEY3D36g1j/wClJol7oDWNx9old4opRGTEVYnDbugwD0q8LXb4m0q2jYMbG0cyH2O1R/I/lRKEFdW7/dbT8SY1Kuj9L+vNZr7vyTOU+OP/ACLOnf8AX5/7I1dT8PP+Sd6J/wBeq1y3xx/5FnTv+vz/ANkaup+Hn/JO9E/69VrxI/7xL0Pua/8AyI6P+N/qdJRRRXWfNhRRRQAV5B8dvvaL9Jv/AGWvX68g+O33tF+k3/stc2K/gs97h3/kZ0/n+TPS/DP/ACKek/8AXlD/AOgCqVz4kkXxHbWFrEj27S+TLK2c78ZIX6d6t+HlZ/BulrG+xzYRBWxnB8sc1g3Gh6vaX2jolzFcLHcF3lS1IwT1Zvm716OEjTkvffT9D5vMp1Y1JezXXp6nRWupyXWsX9ssQ8m0CruHVnIyR+RFVtG1m9v9XvbS+sltPIRHRd+5sMT97tnjtUeiuIde1yCQfvPOWbhfvKUHI/EGq2m32/xhezmzvUiuooo43e1dRld2ckjjqK19mrSSXRfocntZXi3L7TXy1/LQ19O1M3d/f2kiqslpKF+X+JSAQf1rxb4yf8lCh/684v8A0Nq9a0WFm8Va5dj/AFbPHEPcqozXkvxk/wCShQ/9ecX/AKG1eZmUVGNl5fkfYcITlPGty/v/AHJ6HvEf+qT/AHRTqbH/AKpP90U6tjwXuFFFFAgooooA5D4qf8k11T6R/wDoxawfgb/yK+pf9fv/ALTWt74qf8k11T6R/wDoxawfgb/yK+pf9fv/ALTWuOX+8r0Pp6P/ACIKv+Nf+2npFzdQWcJmupkhjHV5GAA/Gh7qCNYjJMiiYhYyWHzkjIA9a5LxPeW2oNf29y+yKxgYqr8CWUjj67f5mtCCSO8vtDUMHjS3MqkH+IAL/U16/wBXtBSfn+V0fFvEt1HCNtLfnZm5Ld28E0cU08cckpwiMwBY+w7043EIuBbmVBMy7hHu+Yj1xXP6/Y2ya5pN6sQ+0SXaoZCSTtCNwPQfSpdcHleJNCnTh2meIn1UoTipVKL5bPdP8Byryi5XWzX3M5340/8AIhp/1+x/yatD4U/8k307/el/9GNWf8af+RDT/r9j/k1aHwp/5Jvp3+9L/wCjGry1/vL9D7Cf/Ihj/wBfP0Z2NFFFdZ82FFFFABRRRQAUUUUAFFFFABWbrPh/T9fjgj1SJpUhfeqhyoJxjnFaVFJpNWZMoRmuWSuipp+l2OlwCLT7WKBAoX5F5IHTJ6n8at0UUJJaIaioqyCobz/jwuP+uTfyqaobz/jwuP8Ark38qHsXD4keK/BO6t7TWdTa6nihDW6AGRwufm969k/tjTP+gjaf9/1/xr5XstOvNRYpYWk90yDLCGMuVHvirf8AwjGuf9AW/wD/AAFf/CvJo4iVOHKo3P0nM8joYzEutOtyt2006L1Pp3+2NM/6CNp/3/X/ABo/tjTP+gjaf9/1/wAa+Yv+EY1z/oC3/wD4Cv8A4Uf8Ixrn/QFv/wDwFf8Awrb65P8AlPN/1Yw3/QR+C/zPp3+2NM/6CNp/3/X/ABrG8Xarp8vgzWEjv7Z3azlCqsykk7T7189/8Ixrn/QFv/8AwFf/AApsvh7WIIXlm0i9jjQFmd7dgFA7k4qZYubVuU1pcN4aFSM1iL2afT/M9T+BX/IP1r/rtF/6Ca9Yryf4Ff8AIP1r/rtF/wCgmvWK68N/BR87n/8AyM6vy/JBRRRXQeGRXP8Ax6Tf7jfyrwf4Ua9pfh/Xr+fWbyO0jkt9iM+eTvBxxXvFz/x6Tf7jfyr5c0Xw9qniO7mg0a1NzLEu91DquBnGeSK4cVKUZwcVrqfYcP0aVfC4mnWlyxfLd7W37n0B/wALJ8If9B23/Jv8KP8AhZPhD/oO2/5N/hXjP/Cr/GH/AEB2/wC/8f8A8VR/wq/xh/0B2/7/AMf/AMVUfWK/8n4M6v7Eyf8A6Cf/ACaP+R66fHXgg6mL/wDtu3FwIzFuG7lc55GParP/AAsnwh/0Hbf8m/wrxn/hV/jD/oDt/wB/4/8A4qj/AIVf4w/6A7f9/wCP/wCKpvE4h7x/BiWR5MtsR/5NH/I9C8U+LfCU/hHXodM1eKa8v4W+X5iXbGABkVn/AAJ/1Wuf78P8mrhL/wCH3ifS9Pnvr7TGit4ELyP5qHaB3wGzXd/An/Va5/vw/wAmqY1KlTERdRWf/ANsVg8Jg8mrQwk+ZNpvVPW67HrlFFFekfABRRRQBU1b/kC3v/XvJ/6Ca8R+EHiLSfD8+ovrN9HaLNFEIy4PzEFs9B7ivbtW/wCQLe/9e8n/AKCa+YvD/hrVvEgkTRrQ3LQKpkAdV2g9OpHpXBiZSjUi4q71Pssho0a+CxFOvLli3G70XfvofQH/AAsnwh/0Hbf8m/wo/wCFk+EP+g7b/k3+FeM/8Kv8Yf8AQHb/AL/x/wDxVH/Cr/GH/QHb/v8Ax/8AxVT9Yr/yfgzo/sTJ/wDoJ/8AJo/5Hs3/AAsnwh/0Hbf8m/wo/wCFk+EP+g7b/k3+FeM/8Kv8Yf8AQHb/AL/x/wDxVH/Cr/GH/QHb/v8Ax/8AxVH1iv8Ayfgw/sTJ/wDoJ/8AJo/5Hp/izx94X1DwhqlpZ6zBLPNbOkcahssxHA6VlfAv/kC6v/19J/6AK89vfh54o06xmvLzS2jggQvI/nIdqjqcBq9C+Bf/ACBdX/6+k/8AQBSpznOvFzVisZhMLhcoqxwtTnTavqn1XY9Uooor0j4MKKKKAOc+IP8AyT7Wf+vZq8l+H3xCs/B2lXdrd2U9w08/mhoioAG0DHP0r1r4g/8AJPtZ/wCvZq8Z8FfDubxnp1zdxailoIJvKKtEXz8oOeo9a8/Ec/tlyb2PtcmjhJZZVWMdocy79lbY7n/heWlf9Am9/wC+k/xo/wCF5aV/0Cb3/vpP8ay/+FFXX/Qdi/8AAY//ABVH/Cirr/oOxf8AgMf/AIqlzYvt+Rp7Lhv+b/0r/I1P+F5aV/0Cb3/vpP8AGj/heWlf9Am9/wC+k/xrL/4UVdf9B2L/AMBj/wDFUf8ACirr/oOxf+Ax/wDiqObF9vyD2XDf83/pX+Rm+PfiTY+LvDyada2Nxbus6y75WUjAB44+tei/Cn/km+nfWX/0Y1eW+MfhpN4Q0RdRl1OO6DTLFsWEr1B5zk+lepfCn/km+nfWX/0Y1FD2nt37TexObRwccpisG7w5/Pez7nY0UUV6J8QYut+FrDxDe2k+pBpEtQwWIHAYkjqfw6e9bEcaRRrHEoRFGFVRgAelOopKKTuiI04Rk5JavcK+ffE1zDZfHGS6upBHDDqEDyOeiqFQk19BV84+PrWa++KmpWlqm+ee4jjjXIG5iigDmuPGaRi13PruF4xliKsZOycH+aPZv+Fk+EP+g7b/AJN/hR/wsnwh/wBB23/Jv8K8Z/4Vf4w/6A7f9/4//iqP+FX+MP8AoDt/3/j/APiqz+sV/wCT8Gd/9iZP/wBBP/k0f8j2b/hZPhD/AKDtv+Tf4Uf8LJ8If9B23/Jv8K8Z/wCFX+MP+gO3/f8Aj/8AiqP+FX+MP+gO3/f+P/4qj6xX/k/Bh/YmT/8AQT/5NH/I9m/4WT4Q/wCg7b/k3+FeY+AbiK7+NM1xbuJIZpLl0cdGUhiDWP8A8Kv8Yf8AQHb/AL/x/wDxVXfhZby2fxQhtrhdk0KzRuuc4YKQRUOpUnOHOranVDAYHC4PEPC1eduDvqn08j6Booor1D87MHxN4Zj8StYR3L7ba3maSVQcM42kAA9ua1dPsLbS7GKzsohFDEuFUfzPqas0VKik+bqZqlCM3US1fU8y+OP/ACK2nf8AX8P/AEW1anwx1Kxt/h3psc97bxSKJMo8qgj943Ymsv44/wDIrad/1/D/ANFtXj9voWq3kCz2ml3c8TfdkjgZlPbggV59So6ddtK+h+g4DA08dk8KdSfKlJu/3n1D/bGmf9BG0/7/AK/40f2xpn/QRtP+/wCv+NfMX/CMa5/0Bb//AMBX/wAKP+EY1z/oC3//AICv/hT+uT/lM/8AVjDf9BH4L/M+nf7Y0z/oI2n/AH/X/Gj+2NM/6CNp/wB/1/xr5i/4RjXP+gLf/wDgK/8AhR/wjGuf9AW//wDAV/8ACj65P+UP9WMN/wBBH4L/ADPQPjheW13JpP2W4in2pNu8tw2Pu+letaD/AMi5pv8A16Rf+gCvlq+02+05QL+yntS4O3zoim7HXGa+pdB/5FzTf+vSL/0AVWGm51JSatsYZ7ho4TAYejCXMk5a/iX6KKK7z40KKKKAOD+Mf/JPJf8Ar5h/9Cqx8Jf+Sc2X/XSX/wBDNV/jH/yTyX/r5h/9CrxOy8S65ptqttp+rXdtApJWOKUqozyeK8+rVVKvzPsfbZdl88wyf2MJJPnb19P+CfVFQzWkE80MssYaSBi0bf3SRj+VfMv/AAmnib/oP6h/3/NH/CaeJv8AoP6h/wB/zTWOitkzN8I4h71I/ifUFQR2VvFeTXUceJpgA75JyB0FfM3/AAmnib/oP6h/3/NH/CaeJv8AoP6h/wB/zTWPitkwfCNd71I/iep/HH/kWdO/6/P/AGRq6n4ef8k70T/r1WvnjUNf1fVoVi1TUrm7jRtyrNIWAPrX0P8ADz/kneif9eq0qFRVKzkuw83wUsDldKhN3ak9vNM6Siiiu8+MCiiigAryD47fe0X6Tf8Astev15B8dfvaL/22/wDZa5sV/BZ73Dv/ACM6fz/Jnpfhn/kU9J/68of/AEAVqV4DY/GHxBp+n29nDa6e0dvEsSFkfJCjAz83tVj/AIXZ4j/589O/79v/APFVnHF0kkjsq8NZhOpKSS1b6nuH2SAXpuxGPPKeWXz1XOcVNXhP/C7PEf8Az56d/wB+3/8AiqP+F2eI/wDnz07/AL9v/wDFVX1ym92ZLhfMFsl957hbWsNnEY7dNilix5JySck814X8ZP8AkoUP/XnF/wChtUn/AAuzxH/z56d/37f/AOKrk/EfiW88V63HqOoxwxzBEi2wghcBie5PrXPicRCrCy3PayPJMVgcV7Wqko2a0Z9QR/6pP90U6mx/6pP90U6vTPgXuFFFFAgooooA5D4qf8k11T6R/wDoxawfgb/yK+pf9fv/ALTWt74qf8k11T6R/wDoxa8q8DfEUeDNLubM6abvz5/N3ibZj5QMYwfSuCrOMMQpS7H2GX4WtislqUqKvLn/AEj3Pf7q2jvLSW2nBMcqlGwcHBqhe6dP9o02WwKj7I+11Y/ejK4I+vArzb/hew/6ADf+BQ/+Jo/4XsP+gA3/AIFD/wCJrpjjacdmeTPhrMZrWl+MfXuenXukRX9xHLNPcAxPvjCPgK2MZ6e5qC50ya58QWNy7A21nGxGT8zSEYz+Rrzj/hew/wCgA3/gUP8A4mj/AIXsP+gA3/gUP/iaax0F9r+mTLhnMJf8uu3VdPmbnxp/5ENP+v2P+TVofCn/AJJvp3+9L/6MavMvGvxMHi/QV00aWbTEyy+YZt/TPGMD1r034U/8k307/el/9GNXNTnGeIbj2PWxuErYTJY0qys+e/4PsdjRRRXcfIhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABUN5/x4XH/AFyb+VTVDef8eFx/1yb+VJ7FQ+JHjXwL/wCQ3qv/AF7J/wChV7XXinwL/wCQ3qv/AF7J/wChV7XXNhP4SPoOJP8AkZT9F+SCiiiuo+dCsXxl/wAiRrP/AF5S/wDoJrarF8Zf8iRrP/XlL/6Camfws6ML/vEPVfmcD8Cv+QfrX/XaL/0E16xXk/wK/wCQfrX/AF2i/wDQTXpuo6rY6TAs2pXMdtGzbFZzgE9cfpWGGaVFNnpcQyUcyqtvTT8kW6Kw/wDhNfDn/QYtv++q4fw38SZLO8ktdbZ7m3knYpcluY1J4yO4q5VoRaTZ8xUx1CnKMXLf+tT0+5/49Jv9xv5V4v8AA/8A5GbU/wDr1/8AZxXsf2q3vdMkntJo54mjbDxsGB49a8c+B/8AyM2p/wDXr/7OKyq/xYfM+wyxp5Zi2u0fzZ7bRRRXWfOBRRRQBzvxA/5J5rf/AF5v/KuF+BP+q1z/AH4f5NXdfED/AJJ5rf8A15v/ACrhfgT/AKrXP9+H+TVxz/3mPp/mfUYT/kRYj/EvzieuUUUV2Hy4UUUUAVNW/wCQLe/9e8n/AKCa8k+A/wDx9av/ANcYf5tXrGrSx/2Xew718w2sjBM842kZx6V5P8B/+PnV/wDrjD/Nq5Kn8eHzPpMB/wAinF/9u/mevRahaTXstnFcRvcRANJGpyVHvQl9bvfS2ivmaJA7jBwoPTnpWFp9pb2fjy7S1iWNWskZsD7xLnJJ7mq97JKmg+I7uElbgysu4dQFUAfzNet7GLlZPdL8T476xNRba2b+5Jv7zftNb02+uDBaXkUso/hU81NZ30F9G72z7wjtG3GCGBwRWNqUMUEeiNaoqyJMix7RztK/Nj2xTtC/d+ItdgT/AFYmjcAdiyc0pUocjlH+tbfqEa81OMJdf1Tf6C+OP+RD1r/rzk/lXE/Av/kC6v8A9fSf+gCu28cf8iHrX/XnJ/KuJ+Bf/IF1f/r6T/0AV5U/94j6H2OG/wCRJX/xR/Q9UooorrPmwooooA5z4g/8k+1n/r2auU+Bv/It6l/1+f8Asi11fxB/5J9rP/Xs1cp8Df8AkW9S/wCvz/2Ra5Jf7xH0PpMP/wAiOt/jX6HcxeIbWfXRpkKSs5Vj5u3CZXGRk9TzUr6kw1s2Sxr5UcHnSylsbOSAMfhWLe6rp6+OrEtfWwEdvMjkyr8rZXg88GrLRvNd69AhzNJCPLAPJBUgfrXrulFJO1rr9bHxka0m5R5tpW07Wvb9CxH4iRpojLZzw2s77Ibp9uxyenGcjPbIq1p2oSXdzeW9xCIpLaXaAGyGUjIb+f5VzU9zDqHhfTdLs5UkvkkgV4FbLxFCNxYdRjFbNoGPjG9ZP9WtrGrY6bsn+lVUpRUXpZ6/g1r8yKdeblHqtPxTuvlZHMfGr/kRY/8Ar9j/AJNWj8Kf+Sb6d9Zf/RjVnfGr/kRY/wDr9j/k1aPwp/5Jvp31l/8ARjV4y/3l+h9vU/5EMP8Ar5+jOxooorrPmgooooAK8C1z/kvg/wCwnb/ySvfa8C1z/kvg/wCwnb/ySuPF7R9T6fhz+LW/69v80e9ySpDG0krqiKMszHAAqK2vba8tBdWsySwMMiRTwabf2dvfWbw3cSyx4yUbofr61yWnyvB8KZHiO1hC4BHbnFepTpKcL31ul99z46pWlTqqNtLN+elv8zqjq9imnJfSTrHbOAVdwRnPTikXV7F7H7ZHcLJb7gpdOQDnHPpWLOir4p0G2bHkR2ztGp6bwAAfrjNVLpEFx4siiwIfsquQOgfaxJ+vArVUIPv3+V7GDxNRdtNPny81/Q7IHIBHINeG+Cv+S5Xf/Xe7/wDZq9i0GZ7jw/ZSy/faFSc/SvHfBX/Jcrv/AK73f/s1eTio8tWMezPschmqmDxM11p3/A9yooorpPnwooooA8y+OP8AyK2nf9fw/wDRbVvfCv8A5Jtpf0k/9GNWD8cf+RW07/r+H/otq3vhX/yTbS/pJ/6MauOP+8v0/wAj6et/yIKf+N/+3HWmRA4QsoY9FzyaZJdQRXEUEkqrLLnYhPLY61zl9ZR2/jrSpw0jyTecSXbOBtHA9BVkv5nim/mIy1nZqsYz0LFiT+g/KvV9jGyd+l/xt+Z8d7aV3G2qaX4J/kbfnw+b5fmp5n93cM/lTYLuC5eVIJVkaFtkgB+6fQ1ySWdu/g2DWPJT+0CFuDcAfOW3Zxu647YrWtH8nxfNGqgC7tEmf/eBK/yx+VOVBJOz2v8AgRHEt8t1o7fc72/E85+PH+t0f/rnN/7JXqmg/wDIuab/ANekX/oAryv48f63R/8ArnN/7JXqmg/8i5pv/XpF/wCgCvKp/wAefyPscd/yKML6y/Mv0UUV1nzYUUUUAcH8Y/8Aknkv/XzD/wChVW+GHh/R7/wFZ3F9pVnczM8gaSWBWY4c45Iqz8Y/+SeS/wDXzD/6FU3wobZ8NbNsFsPKcDqfnNclk8TZ9j6ZTlDIk4u37z/203/+ES8O/wDQC07/AMBU/wAKP+ES8O/9ALTv/AVP8KbBrd1/aFvb6hpjWiXORFJ5ofkDOCMcHGam/tqJZb9pNq21iAJJM87sZIx+X513PDtdPy9D5tY+b/5eP736kf8AwiXh3/oBad/4Cp/hR/wiXh3/AKAWnf8AgKn+FR23iCR7q2jvbB7SO8H+jSGQNu4yARj5SR2q7peo/bxcJIgjmtpmikUHPTofxGDRLDuKu1+Qo5hOTsqj+9+v5HmXxj0XS9M8PWEmnada2jtdbWaCFUJGw8ZArt/h5/yTvRP+vVa5b44/8izp3/X5/wCyNXU/Dz/kneif9eq1wwSWIlbsfS4qcp5JScnd87/U6Siiius+ZCiiigAryD46/e0X/tt/7LXr9eQfHb72i/Sb/wBlrmxX8FnvcO/8jOn8/wAmWtI+Dmh6holjeTX2oLJcW8crBXTALKCcfL71c/4UhoH/AD/6l/32n/xNdr4aOPCOknBOLGHgf7gqBtfuLfULaG+0yS3gupPLim8wMd3bcuOM1pTwkKi0iZYnPcdSqSTqtK7/AD9Dkf8AhSGgf8/+pf8Afaf/ABNH/CkNA/5/9S/77T/4mu3XWFOqXtuVAgsog8sxPRjzj8ufxqpa+JHlmtftenyWtteNtt5mcHce24Y+XParWBi/snO+IsanZ1n/AFp2OT/4UhoH/P8A6l/32n/xNea+PPDNp4T8VRadYSzSxGGOXdMQWyWI7AelfQunaj9tluoXUJNaymN1BznuD+IIrxP4yf8AJQof+vOL/wBDauLF0I0oaKzPo+Hc0xWMxfJUqOUbP70e8R/6pP8AdFOpsf8Aqk/3RTq7z4x7hRRRQIKKKKAOQ+Kn/JNdU+kf/oxa8s8CfDqPxnpVzePqTWhgn8rasIfPyg5zketep/FT/kmuqfSP/wBGLXPfBJ/L8J6o4Vn23hO1Rkn92tcNSEZ4hRl2PsMBiquFySpVou0uf9I9yt/woqH/AKD0n/gKP/iqP+FFQ/8AQek/8BR/8VXoEGuzjUbe11HTms/tQPkN5ofJHJBGODipBrQWXUXnQR2tkQvm55c4BIx+OK63gIL7P4/LueKuJ8e/+X3/AJKu1+3Y87/4UVD/ANB6T/wFH/xVH/Ciof8AoPSf+Ao/+Kr0C01y4kvILfUNOay+0qTCxkDBiBnB44OO1WdH1M6nbzGSLyZoJmikTOcEdD+IwaUsBCKu4/j/AMEceJsfJpKr/wCSr17Hifjf4Zx+ENAXUk1R7omdYvLMITrnnOT6V6X8Kf8Akm+nf70v/oxqz/jT/wAiGn/X7H/Jq0PhT/yTfTv96X/0Y1ctOEYYhqPY9fG4qti8ljUrSu+f9H2OxoooruPkQooooAKKKKACiiigAooooAKKKKACiiigAqG8/wCPC4/65N/KpqhvP+PC4/65N/Kk9iofEjxr4F/8hvVf+vZP/Qq9rrxT4F/8hvVf+vZP/Qq9rrmwn8JH0HEn/Iyn6L8kFFFFdR86FYvjL/kSNZ/68pf/AEE1tVi+Mv8AkSNZ/wCvKX/0E1M/hZ0YX/eIeq/M4H4Ff8g/Wv8ArtF/6Ca3fiypbw1abVJ/0sdBn+BqwvgV/wAg/Wv+u0X/AKCa9XIDdQD9a5qMefDqJ28UUfb42tTva9vyR81+W/8Acf8A75NNr6V8tP7q/lXH+Hfh7Z6XeTXmp+Xe3DSl4vl+WMZyOD1NYywkk0kz4CpktRSioSvffpYyPh9oOs2NndXV5LNZ2boStqyj96dp+bn7o5/Gub+B/wDyM2p/9ev/ALOK9ouf+PSb/rm38q8Q+DV9aWPiPUnvbmG3VrbCtLIEBO8cc1UoKnUpr1P0fI8MqOU4qlC70j+bPdKKzv8AhIdG/wCgvY/+BKf40f8ACQ6N/wBBex/8CU/xrt5o9zx/YVf5X9xo0Vnf8JDo3/QXsf8AwJT/ABo/4SHRv+gvY/8AgSn+NHNHuHsKv8r+4zviB/yTzW/+vN/5VwvwJ/1Wuf78P8mrrPHWt6VceAtZig1Kzlke0cKiTqSxx0ABrk/gT/qtc/34f5NXJNp4mNu3+Z9LhoShkeIUlb3l+cT1yiiiu0+VCiiigCnqyj+x71sDd9mkGcc42mvJfgPzc6v/ANcYf5tXrerf8gW9/wCveT/0E1458Eb+zsbjVDfXUFsHiiCmaQJu5bpmuSrpXh8z6XLouWU4pJfy/mesReGrCG+F4jXXnjHzm6kOQDnB55HtT7bTZYNXv5GKPZ3YVjG3VXAweOmCKd/wkOjf9Bex/wDAlP8AGj/hIdG/6C9j/wCBKf413OvJ7yufNfUmrWg1rfYSy0GxsJ1mgWRmQEJ5krOIx/sgnj8KTRtNlsTdzXLK091O0rbegHRR+AAp3/CQ6N/0F7H/AMCU/wAaP+Eh0b/oL2P/AIEp/jQ67ad3uCwck1aDVvL5FLxx/wAiHrX/AF5yfyrifgX/AMgXV/8Ar6T/ANAFdR4z1vSp/BOrxQanZySPaSBUSdSWOOgGa5f4F/8AIF1f/r6T/wBAFcMmniI27H01CEoZLX5lb3o/oeqUUUV1nzIUUUUAc58Qf+Sfaz/17NXKfA3/AJFvUv8Ar8/9kWur+IP/ACT7Wf8Ar2avMfhl460Xwpo97bau86yTXHmJ5cJcY2gdvpXHUko4iLfY+qwNGrXyarClFyfOtF8j2w2tuWLGCIknJJQVAdNj/thdRV3WUReUyjG1xnIz9K4//hcfhL/ntd/+ArUf8Lj8Jf8APa7/APAVq6FiYLaR4rybHPehL7mdysMSSF1jRXPVgoyarWOnR2MlzIsjyPcy+Y7Pj0wAMdhiuP8A+Fx+Ev8Antd/+ArUf8Lj8Jf89rv/AMBWo+swtbmH/Y+ObT9jLTyZD8av+RFj/wCv2P8Ak1aPwp/5Jvp31l/9GNXD/Ej4gaF4o8LpYaTJO063KSESQlBtAOeT9a7j4U/8k3076y/+jGrmhKMsQ3F9D2sVQq4fJIQqxcX7TZ+jOxooortPlAooooAK8C1z/kvg/wCwnb/ySvfa8A8QSJD8dzJK6oiajbszMcBRtTkmuPF/DH1PqOG9atb/AAP80e73tlHfweTM0qrnOYpWQ/mDVGHw3YWul3FhbecsE8ZQq0rOFB9ATxUv/CQ6N/0F7H/wJT/Gj/hIdG/6C9j/AOBKf413Ks4qylofMvCSlLmdPX0GR6OtzpNrb6oN81uoAlicqwI4yGGCM0260VBos9hYKE+0fLK7sSxB+8STkk49al/4SHRv+gvY/wDgSn+NH/CQ6N/0F7H/AMCU/wAaft3e9/Mj6k7W5HtbbUvQRLBbxxRjCxqFH4V4h4K/5Lld/wDXe7/9mr2H/hIdG/6C9j/4Ep/jXjngaRJfjdcyRMro010VZTkEfNyDXDiJJzh6n0+T05U8Lik1b3P8z3Siiius+ZCiiigDzL44/wDIrad/1/D/ANFtW98K/wDkm2l/ST/0Y1YPxx/5FbTv+v4f+i2re+Ff/JNtL+kn/oxq44/7y/T/ACPp63/Igp/43/7cat54eku9Ujvjqt3HJCW8pVWPCBuoGV/nUjWVxD4mS5jXzLee38qcnHyspJU++ckVrUV6XtpWs+1j5F0IXut7p/d/VvQxU8NRJIFN5ctaLL5q2hK+WpznHTOM9s4qWxtbhtevr26h8tcLDb8gkoOSePUmtWik602ncaoQVrHjnx4/1uj/APXOb/2SvVNB/wCRc03/AK9Iv/QBXlfx4/1uj/8AXOb/ANkr1TQf+Rc03/r0i/8AQBXn0/48/kfVY7/kUYX1l+ZfooorrPmwooooA4P4x/8AJPJf+vmH/wBCqb4UuI/hpaO2dqtMTgZP3zUPxj/5J5L/ANfMP/oVWPhL/wAk5sv+ukv/AKGa5F/vPyPo3/yIV/18/wDbR6yWMmt2M/h+7kvbjzMTI8jShIj95vm+7jjpiovsc91oPiS0hQvc/bS20HluEP8AIV2oVR0AH0FQRWMUN9NdR7leZQHUH5SR3x617CxNtltbfydz414Vvd77203TXmc3Pdxa3NosGnHzXt5kmuAB/qQqkEN6HPGKuaEjHxJr0o/1TTooPbIQZrf2gZwMZ9Kr2NhFp8LRw7jvdpHZzlmYnJJqHWTg4peX43KWGkpxk3s7/cmv1PO/jj/yLOnf9fn/ALI1dT8PP+Sd6J/16rXLfHH/AJFnTv8Ar8/9kaup+Hn/ACTvRP8Ar1WvKj/vEvQ+xr/8iOj/AI3+p0lFFFdZ82FFFFABXkHx2+9ov0m/9lr1+vIPjt97RfpN/wCy1zYr+Cz3uHf+RnT+f5M9L8Nnb4R0pjnAsYTwM/wCsG6fT7jWLKfRrqa5vRcAvA7NIEUn5iVb7mPbFb3hn/kU9J/68of/AEAVphQDkAA/Su3D1PZq/keDjqXtasl5v8+hyqQtcnxTZxc3EjkqoPJBiGKilvYtZtdHs7HL3ENxFJcJjBgCdd3pXTJp8EeqSX6ArNJGI3weGAPBPvVkKAcgAH6V0fWIp3S7ffaxw/VZO93vf7m7nPaMjnxhrswH7rdGme24IM15P8ZP+ShQ/wDXnF/6G1e42VjFYpIItzNI5kd2OSxPrXh3xk/5KFD/ANecX/obV5mYzU4XXkvuVj67hGm6eNafVSf3u57xH/qk/wB0U6mx/wCqT/dFOrc8B7hRRRQIKKKKAOQ+Kn/JNdU+kf8A6MWud+Cciw+EdVkk3bVvCTtBJx5a9hXRfFT/AJJrqn0j/wDRi1g/A3/kV9S/6/f/AGmtccv96XofT0f+RBV/x/8AyJuLJaya9YTeHbiS+cuRch3aURxnqct9z8OtPuLaW78MatawDzLmO6LMgPJwwbH5V1oUDoAPoKqwadDbahc3cRYPc4Mi5+UkDGceuK9j6wt+1rfffU+KeFeqvvv06W03MSa+tte1DSk0yUTC3l86ZlHEYCEYPoST0qx4d+bVdclT/VtdgKfUhBmt0KBnAAz6Cq2nadBplqYLbdguzszHJZickmodWPI4peX43KVCXtFNvzf3W/U4j40/8iGn/X7H/Jq0PhT/AMk307/el/8ARjVn/Gn/AJENP+v2P+TVofCn/km+nf70v/oxq8tf7y/Q+wn/AMiGP/Xz9GdjRRRXWfNhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABUdxGZbWWNeroVGfcVJRQNOzueEw/B3xZbkm3vLSIkYJjuXXP5LUv/Cp/Gn/QTh/8DZP8K9xork+qU/M+kfEuOer5fuPDv+FT+NP+gnD/AOBsn+FH/Cp/Gn/QTh/8DZP8K9xoo+qU/MX+smN7R+48O/4VP40/6CcP/gbJ/hTW+EvjJ1KvqMDKRgq15IQf0r3Oij6pT8w/1kx3aP3HDfDPwdqXhC01CPVHgZrmRGTyXLYABHOQPWu5oorohBQjyo8TFYmpi60q1Td/8MFFFFWcwyVPMhdAcblIzXi7fAzVGY/8TizxnjMLV7XRWVSjCpbmPRwOZ4nAc3sHa++ie3qeJf8ACitT/wCgvZf9+Wo/4UVqf/QXsv8Avy1e20Vl9Uo9j0v9Zcy/nX3L/I8S/wCFFan/ANBey/78tR/worU/+gvZf9+Wr22ij6pR7B/rLmX86+5f5Hif/CitTHTWLL/vy1dx8O/A914Ljvxd3kN0bpkI8pCu3aD6/Wu0oqoYenCXNFHLis7x2LpOjVleL8l6hRRRXQeMFFFFAEVzALq0mgYlVlRkJHbIxXkd/wDAlg0Y0rWAUA+b7XFk59ttew0VlUpQqfEj0MFmWKwLfsJWvvomeJf8KK1P/oL2X/flqP8AhRWp/wDQXsv+/LV7bRWX1Sj2PT/1lzL+dfcv8jxL/hRWp/8AQXsv+/LUf8KK1P8A6C9l/wB+Wr22ij6pR7B/rLmX86+5f5HiX/CitTH/ADGLL/vy1d98PfBlz4MsL2C7u4rprmYSAxKVCgLjHNdfRVww9OEuaKOXFZ1jcXSdGtK8X5IKKKK3PGCiiigCrqWnW+rabPYXyF7e4QpIoYjI+orlP+FSeEf+fGb/AMCX/wAa7WiolThLWSudVHGYnDrlozcV5No4r/hUnhH/AJ8Zv/Al/wDGj/hUnhH/AJ8Zv/Al/wDGu1pAQc4IOOtT7Gn/ACo3/tTH/wDP6X3s4v8A4VJ4R/58Zv8AwJf/ABo/4VJ4R/58Zv8AwJf/ABrtaMgnGeRR7Gn/ACoP7Ux3/P6X3s4r/hUnhH/nxm/8CX/xrp9H0ez0HS4tP02Mx20WdisxYjJJPJ9zV6iqjThF3ijGtjcTXjy1ajkvNthRRRVnIFFFJkbsZGfSgBa8q8V/CTUPEHim+1WDVLaGO6ZWEbxsSuFA7fSvVaa0qK2GdQfQms6lONRWkd2Cx2IwVR1KDs2rbXPFP+FFan/0F7L/AL8tR/worU/+gvZf9+Wr2S+1Gz0yATahcxW0TNtDysFBPpVhHWRA6EMrDII71h9Uo9j1v9Y80S5ubT/Cv8jxP/hRWp/9Bey/78tR/wAKK1P/AKC9l/35avbCQoyxAA7mlBBGQcg9xT+qUewv9Zcz/nX3L/I8S/4UVqf/AEF7L/vy1dD4K+Ft94W8UQ6pc6lbzxxo6+XHGwJ3DHevTKKccLSi7pGNbiDMK9OVKclZqz0QUUUV0nhBRRRQBxvxJ8KX/i3RLS00toVkhufNbzmKjG1h2B9a89j+EfjGJAkWoW8aDoqXcgA/ACvdKK554eE5czPawmd4rCUVRp25V3Vzw7/hU/jT/oJw/wDgbJ/hR/wqfxp/0E4f/A2T/CvcaKj6pT8zq/1kxvaP3Hh3/Cp/Gn/QTh/8DZP8KP8AhU/jT/oJw/8AgbJ/hXuNFH1Sn5h/rJje0fuPCJ/g94tuV/f3lpLgEDzLl2x+a17bpls9npNnbSkF4YEjYr0yFAP8qtUVrToxptuJ5+OzXEY+MY1rWjtZWCiiitjywooooA4j4uW0914AljtYJJ5PtER2RIWON3oK8q0jxL440LTUsNLhvIbaMkqh08tgk5PJX1r6MormqUHOfOpWPfwOcQwuG+r1KKmr31/yszwD/hPfiN6Xf/gs/wDsaP8AhPfiN6Xf/gs/+xr3+io+rz/nZ0/25hf+gSH4f5HgH/Ce/Eb0u/8AwWf/AGNH/Ce/Eb0u/wDwWf8A2Ne/0UfV5/zsP7cwv/QJD8P8j5s13WvGXiW1jt9Ztr2eKJ96KLAphsYzwvoa9y8AxSQeANGinjeKRbVQyOpBU+4NdDRV0qDpycm7nJmGbRxlCNCFJQSd9P8AhkFFFFdJ4QUUUUAFeQ/HVSW0XAJ/1v8A7LXr1NaNH++itj1Gayq0/aQcTvy7GfUsTHEct7X023VjyPSvjNaado9nZPo1y7W0CRFhKoDbVAz09qt/8Lysv+gHdf8Af5f8K9P8iH/nkn/fIo8iH/nkn/fIrNU6qVuf8DuljctlJyeGd3/ff+R5h/wvKy/6Ad1/3+X/AAo/4XlZf9AO6/7/AC/4V6f5EP8AzyT/AL5FHkQ/88k/75FPkrfz/gL65ln/AEDP/wADf+R5h/wvKy/6Ad1/3+X/AArzzxr4lTxd4ni1KC0ktlESQ+W53HIYnPH1r6S8iH/nkn/fIo8iH/nkn/fIqJ0KlRWlL8Dqwmb4LB1Pa0cPZ7fG3+g6P/VL/uinUUV1nzIUUUUAFFFFAHIfFT/kmuqfSP8A9GLXm/w4+IOleENGu7TUobqSSa481TAgIA2gc5I9K9wvLK21C0e1voI7iCT78ci5Vuc8isn/AIQnwx/0AbD/AL8LXLUpTdRTgz6DA5jhaeClhMTBtOV9Hbt/kcr/AMLu8Of8+mo/9+l/+Ko/4Xd4c/59NR/79L/8VXVf8IT4Y/6ANh/34Wj/AIQnwx/0AbD/AL8LT5cR/Mg9tk3/AD6n/wCBI5X/AIXd4c/59NR/79L/APFUf8Lu8Of8+mo/9+l/+Krqv+EJ8Mf9AGw/78LR/wAIT4Y/6ANh/wB+Fo5cR/Mg9tk3/Pqf/gSPLPiD8SNI8WeGl0/ToLuOUXCS5mRQuAD6E+td/wDCn/km+nf70v8A6MatT/hCfDH/AEAbD/vwta1lY2unWiWthbx28CZ2xxrtUZOelFOlNVOeb6CxmYYSeCWEw0Gkpc2rv0aJ6KKK6TwAooooAKKKKACiiigAooooAKKKKACiiigAooooAwdU1TVk1tdO0eCzdvs5mJuXYZ5xgYFZt74zkXRdPlhWG1vLu7+zSJOwIhwTubqMj5f1qTVbS8vfGqx2GoNYMbFg0ixK5I3DpnpVHxHoNjp9v4ctBCs4XUQrvModnyrk5J9TzXKnUbv0vb/ya35HuUKeGfs4zSu1fr2e/TsdtbktboWlWYkZ8xBgN7jk1leJ9Zu9E0eW60/T2vZkUtt3BUUDkljWTrGjarJqTnTbe4FsAoQQ6u1uoAHQIEIFS61cx6Z4Cnt9VuEt7h7Z1VJ7rzGY4PG44LH8KqpUfs5NaWOalh4e0py0ldrT/hnf8joNLvG1DSLS8dQjXEKSlQeBuAOP1q1WL4SvrW88L6eLS5inMNtEkgjcNsbYODjoa2q3TurnFXhyVZRtazCiiimYhRRRQBFczi2tZZ2BYRoXIHfArndP8aJdCxkv9NuLC31AhbWaR1cSMeg+Ukrn3rc1b/kD3n/XF/5Vyfh3QptY8N+HZb+4QWlnEk0cMSEF2A43Ent7VhKU/acq8vzPRw1Og6Mp1e9ut9m9Pnbc37DU76/1y9hSGBbC1cReYWbzHbAJwOmOa2K47TLb7LFr02nRypcpdFUaBBI4GB0Vjg/jU2i3mty6tEt8+pmA53faLCKNOndlYkU41NFdbpDrYaLvKDSUUtNbvS/mXD4nkt9Qgh1PSrixt7mYww3MjoQ7c4BUEkZAPWt+uTkl/wCEq8RxIiqNK0ucSifdn7RMARtA9Fycn1FdZVU22rmOKhCHKkrStqvy+dtwooorQ4wooooAKKKKACuY1MXl74oktItWubCCK0Ev7jZjOTydyn0rp65+50Zb/wAWyzXtp5tobQIGb7pOTkVlUTbil3/RnXhZRjKUpduyfbozm73XLvVNA0KK7aZlvL4xzyWqsvnIpbGCORnaDxXaw32m2MEdubuGDy1AEc0w3qMd9xzn61l6/pUj3WgLp1qPItL0O6xgBY02MM49ORRqfhV9R1GW6F3axiTHyyabFKRxj7zDJqI88U+rv+iO2pLD1YxTfLH3nb5+S7FrWNlxDFcf2+2nWwBO+FkHmf8AAmBGPoKTwnfXmoeH4p9QyX3sqSEYMqAkK5HuMGq16gsLe1sbzQzrFvHHxJBbxna2f+eZwFGMdKl8Iafe2Glzf2huj8+4eWK3Zt32dCeEz7enarjdzZjOMFhXqt1bbbXtr63N6iiitTzQooooAKiuYTcW0kIkeLepXfGcMvuKlqG8a5SylaxjSS4Cny0dtqlu2T2pPYqN+ZWOTisRovjbTbPR553jlgkN7HLO0nAxtY7icHOemM1en8XNFDLcx6RdS2lvIyXE4ZQIwrYLYJyw78VV8NWGuabdSS6jpds9xeSbrq8F7ubHYBdn3RngZq9/Yt3/AMIrqen/ALvz7oz+X83Hzk4yfxrmvNR93zf5WR7FV0nUSqtStZXvvq7vR9Ol+lie/wDES295FZadaS6jdyR+d5MTKu2P+8WbA69utZnge4+1XHiCbY8e7Uj8jjBX92nFPbRdT07V4NV02OG6lNmtrNbyS+WBt5DBsH8sVb8L6Tf6Y+qyamYmkvLwzr5R4wVUY/SrtJ1b+v56fgZy9hDDSUGtUuurd9dPL+r9JL7xC8OpSWWmabNqU0ChrgROq+VnoDuIySOwrOGp3Nn4h1VrTTLi+dkhYxxsq7RhupYgZ9qnk0/VdL1++v8ASbaC9TUNhdZZ/KMTKNvHByD+FaFjYXEOrahdTbNtykYXae6g5/nUvnk0+uv6k3o04aJNOK66t3je6vpZ3ttp3EXWzcaNDfafYz3TzDiBSFKnuCTwMVVTxXD/AGXf3NzaTQT6ccXFqSGZeMjBHByCDxWZN4e1mLTdPS2aOX7PcSPPaeeY1nVjxlgO3pimW/hfURp+vK8NvA+pFGiiSUsEwgBBJHqOtNyqa27GkaOEs7yW/fX4rW37a7fPobdt4hkk1aOzvNMuLNJ1Jt55WUiQjtgHIOOefSmWPiU6nd/8S/Tbiew8wxm9VlC5HX5SdxGe+Kn1LTZ7vUdLmj27LWUtJk84KkcfnWd4f03WNBiTSkgt57FJGIuzMQwUnONmOv41V5J2f9bf8ExUcPKm5JLmttf1u999tPPYf/wmCEtcrp1wdKSTy21DcuwHOM7c7sZ4zippxu8dWTL0+xSEn/gS1k2Phi6sV/s2TSrK9sxIcXUs7AlSc8x45I+tdE9pP/wkUF0sEZhSBo2k8whhkg4C9McdaI8zab7/AKMur7CnJ+y7Pruradd/u9DSrz3U7jSo/G2uRatGJHlsolgXyS7bvn+6QDg9OeK9CrLtNHNt4lv9VMwYXcUcYj2/d255z+NKrBzcbf1ozDB1oUedy6rS2mt099TjYLXxFBH4Zt7xbV5o2kKrduxYHnG7AIyFIrt31QROY2s7xivBMcBKn6GkvdMN3qlhdiUKLRmYrtzuyMU6XRrCaVpJbcM7HJO48n86qMZJO3c0r4inX5XNW327ttmD4rmfWIv7FhjbyZLdp71j8pSMAlV+pYYx6Zq/4IJPgXRiev2RP5VZu/D2nXUc3+jIsskfl+ZznAGBnnnFL4e0ZNB0K109GEjQxhGkAI3kd8dqUYyU3JhUr0nhFSjumnb5O7/JfcadFFFbHmhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAZ2vaxHoOi3GozRPMkC5KIQCfzqrp3iM3V69nf6fNp9yITOkcjK/mIOpBUkdT061U+If/Iiajj+4P509dDnmuZdS1W5SbFmYYoIkKqinBJznJJIrmlOftGlsv+CenSpUPqynU3ba6305bW6ddbljw1q19rlj/aFxBBDaTZNuqMxfbkgFs8dPSr+raimk6Tc38qNIluhcqvU4rjvDy3Nn8O9Mm0kXKTyAeYbWBZmI56hyAB71oWRudRs7228Sy3y2ckQRjd20duOTjhlY81TlLl5Vvbc0rYWCrSkmuVStbW9k7f1qaujaze6r5ck2jT2VvLEJEmkmjYHOCBhST0NbFcbDHL4d8W6Xpml3d1eWd1C4mhnmMvkKmNrgnoOcYrsq0g21qcmLpxhJSgvdkrrfu1rdvsFFFFWcYUUUUAFRXLyx2sr20fmzKhKITjcewzUtQ3d1DZWc11cuEhhQu7HsAMmk3ZXZUbuSSVzEs9Y1eHXoNO1y1tY1uoWeKW2dmwwIyrZ9j1rZN/ZiRUN3AHY4VfMGSemAK5Hw7rFlr+vHWLjU7cO6NBZaeJVLIueWIzkscZ9hWWmmWb/DzXLxrWM3UdxdPHMV+dCshIKnqPwrm9s4xvvu/kj2Z4OEpqM/dfup2Wl3fv26/geiz3VvaqGup44VY4BkcLk/jWT4c1mfV5dUE3l7LW8eGIxjqo6E+tYdxLYf8JNay+JvLNo+mp5DXQzEZc/N143Y/HFT/D4QCPWvsi7YP7Rk8sYx8uePwrZSbnb1/NGMsLGnhpTer01tpq+j/M6qe8trUqLm4ihLfdEjhc/TNZUGugeItTsr2aCCC1SExM7BSS4YnJJ56CsMtpaeMtZPinyAjeWLP7bgqU2jOzd33ZzjmprTS9O1Lxbrwu7SK4iFtbhBLHnAKv0B6Vm6jtdf1uOOFpwjJzv8Kd7d3Hb77M62W4hhh82aWOOP++7AD86Rbu3e389J4mh/56BwV/PpXA6fqcUHg3QodSitpY5pJYzPfkmKLaWxnPU44GfSqlyUbwl4qig8n7MJVMS26FYgCiklAexPPHWk61ldLpcqOWe9yt9bXtpbm5T0iK7tp5GjguIpHT7yo4JX6gUfbbUXP2c3MPn/APPLzBu/LrXKX1hbaZ4u8OvptvHbPP5qSmJdvmDyyQGx1555qHw82iCHZe/Zjrv2iQkTAG48zJxj+LH6YqlUd7Pp/Whi8HDk9pFtpq+2u7Wvlp+J2P221+0/Z/tMPn/88vMG78utZj6tcL42h0kBPs8lk85OPm3BgOvpzXF6XbxXGmxW+oXmnWmtefmZpIybsSb89c5ORjHbFdW6J/wseGTzo940508rJ3/fU56Yx+NLnlJx9f0ZpPCU6MpRbvpLp26ryOioooroPJM7U9Ru7Jo0sdKn1BmyW8t1QIPcsQKXRdYg1vTVu7dWT5ikkb/ejcHDKfoar+IdcOj2qJaxLc6hcNstrbdjee5PoB1JrDu9IbSvCdnZJdyCae/ia4mgcoxZ5QWwR9TWLm03bX/M9GnQhUpR5lZt6Putbu3lpbY7SsrxJq76Hocl9FEsrI6LtY4B3MF/rWRco+i+IZF0wyt5umyv5TOWDOn3Tg9zmuX1RbWf4a2V/c38hv57hC7NMcyv5vKEeg9PaoqVWou39a2N8LgYyqQlJ3i2une++umx6mpyoPqKq6reNp+k3V2iB2hiZwpOAcCq3iC5uLPwtfXFiCbiO3Zo8DJzisK6stPh8JtPFdO1xJYMcmcnziUySRnn1q6s2lJLov8AP/I5MPh1PlnLZu1v62On0u7bUNIs7x1CNcQJKVB4BZQcfrVqvPo7QaVpHhC7s551lllgilJlJEiunII7jjioPFOpv9k1m502BgbScRteSXxR45ARwiY6e2Rmk6ySd1t/kn+p1f2d7Sqo03o392tu/wDwfI9Iorl0tv7V8TQ/a5pjGtgshiWQqjsT1IHWsSe6uINJttPF1Mlr/bJtZJTIdwiySFLehOBT9pra3f8AO36mMMFztJS10vp3v9+x6HSOSEYjqBxmuJ8WhdOtLeDSJSEuL2NbmP7WUVVwcDdzsDEAe9XtAj1CHV70XUdvaWrW6sttHeecVbJy3IGAR/Km5vVf1sS8Hal7Xm+XXp53NnQtSfVtHivJY1jZycqpyBg4rQrG8JgL4atgpU4BztYEZye4rZrSOyObERUa0lHa7CiiimYhRRRQAUUUUAFFFFABRRRQAm0bt2BnpnFDKrY3KDg5GR0paKACmSQxy482NXx03KDin0UBewyOKOIERRqgPXauKfRRQG4UUUUAFFFFADZI1ljaORQyMMMp6EU23t4bS2jt7WNYoY12oiDAUegqSii3Ud3axFDbQ27SNBEsZlbe5UY3H1NSkZGD0oooBtt3Zix+D/DsUgePR7RGDbgRGBz1zW1RRSUUti51alT45N+ruFFFFMzCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAhu7S3vrZ7e8hSeF+GjcZB/CpdilNmBtxjHtS0UrIfM7WIrW0gsbZLezhSGGMYSNBgKPpRdWsF7bPb3cSzQyDDI4yGFS0UWTVmPmlzc19Slp+jabpO/+zbKG23/AHvLTGau0UUJJaIJSlN80ndhRRRTJCiiigApGUMpVgCD1BHWlooAiW1gRgyQRqw6EIAaf5abSuxdp6jHBp1FA7tjHijkUCSNWA6BlzinKipnYoXPXAxS0UBdjHhilIMkaOR03KDinBVDEhRk9TjrS0UBdjGhiZAjRoUHRSoxS+VGFK+Wu09Rtp1FAXYhVSQSoJXocdKb5EXmeZ5Sb/720Z/On0UBdjDBEZPMMSb/AO9tGfzp2xd+/aN2MZxzS0UBdhRRRQIztR8P6Tq06Tanp9vdSxrtR5UyVHoKdbaFpdnb+Ra2MEUQkEuxU43jo314FX6Knlje9jX21XlUeZ2XS5E1rA10tw0SmZFKrIRyAeozVF/DejSTyzPpls0kxBkYxjLcg8/iBWnRT5V2FGrUj8MmhCoK7SMjGMVm2/hzRrVpmttNtozMpWTbGPmB6g1p0UOKe4o1JxTUW1crNp1m8NvE9tGY7Yq0KleIyowCPTFVbjw3o13eSXVzpltLPIMPI0YJbjHP4Vp0UcqfQca1SOsZNfMhS0t4phLHCiyBBGGA52joPpUT6XYyW01vJaRNDOxaVCow5PUmrdFFkxKck7pmfFoWlQafJYxWECWsuS8IQbW+oqSw0jT9MheLT7OG3jkOWWNcBquUUcq7DdWo005PXzIbW0t7G3EFnCkMSkkIgwBmpqKKZDbbuwooooEFFFFABRRRQAUUUUAFFFFABRURurdWIaeMEdQXHFJ9rtv+fiL/AL7FK6K5Zdiaioftdt/z8Rf99ij7Xbf8/EX/AH2KLoOWXYmoqH7Xbf8APxF/32KPtdt/z8Rf99ii6Dll2JqKZHNHLnypEfHXawOKfTFZrcKKKKBBRRWFceNvDNrcyW9zrdlHNExR0aUAqRwQaTko7s1p0alV2pxb9Fc3aK57/hPvCn/Qfsf+/wAKP+E+8Kf9B+x/7/Cp9pDujb6liv8An3L7mdDRXPf8J94U/wCg/Y/9/hR/wn3hT/oP2P8A3+FHtId0H1LFf8+5fczoaK57/hPvCn/Qfsf+/wAKv6X4i0fW5JE0jUbe8eIAuIX3bQfWmpxbsmRPC4iEeacGl6M0qKKKo5wooooAKKKwZfHHhiCZ4ZtcskkjYq6tKMqRwRSclHdmtOjUq6U4t+iub1Fc9/wn3hT/AKD9j/3+FH/CfeFP+g/Y/wDf4VPtId0bfUsV/wA+5fczoaK57/hPvCn/AEH7H/v8KP8AhPvCn/Qfsf8Av8KPaQ7oPqWK/wCfcvuZ0NFc9/wn3hT/AKD9j/3+FXtL8R6PrcskWk6lb3jxrudYX3FR60KcW7JkTwuIhHmlBpejNOiiirOcKKKKACiq9/eJp2m3N7MGaO3iaVgvUhRk4/KvPR8b9BIz/Z+of98p/wDFVnOpCHxM7cNgMTik3Qg5WPSqK81/4XfoP/QP1D/vlP8A4qj/AIXfoP8A0D9Q/wC+U/8Aiqj6xS/mOr+xMx/59M9KorzX/hd+g/8AQP1D/vlP/iqP+F36D/0D9Q/75T/4qj6xS/mD+xMx/wCfTPSqK81/4XfoP/QP1D/vlP8A4qut8J+LLPxfpst5YQzQpFL5TLMADnAPYn1qo1qc3aLMK+WYzDQ9pWptI3aKKK1PPCiiigAoorK1LxRomjXQt9V1S2tJiu4JLIASPWk2lqy6dOdR8sE2/LU1aK57/hPvCn/Qfsf+/wAKP+E+8Kf9B+x/7/Cp9pDujo+pYr/n3L7mdDRXPf8ACfeFP+g/Y/8Af4Uf8J94U/6D9j/3+FHtId0H1LFf8+5fczoaK57/AIT7wp/0H7H/AL/CpbXxp4bvbqO2tNas5p5WCxxpKCWJ7Cj2kO4ng8Sld05fczcoooqzlCiiigAopHdY13OwVR3JxUX2u2/5+Iv++xQNJvYmoqH7Xbf8/EX/AH2KPtdt/wA/EX/fYpXQ+WXYmoqH7Xbf8/EX/fYo+123/PxF/wB9ii6Dll2JqKh+123/AD8Rf99ipVZXUMjBlPQg5Bpiaa3FooooEFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRTWkRDh3VT7nFADqKZ58X/PVP++hR58X/PVP++hQK6H0V5lL8S7zSvEl9aXsCXVpHcsqsnyuig447Hv/AI12+jeJtJ16Mf2fdo0m3LQtw6/gfr1HFZRqwk7JnJRxlCtJxi9V0NaiiitTsPm5fD0/in4k6lpdtcJbySXVw4kkBIG1ie1dL/wo3Vv+gzZ/9+nqHwP/AMlyvP8Arrd/zNe5V5tChCom5dz7rNs4xmCqwpUJWXKnsn3PE/8AhRurf9Bmz/79PR/wo3Vv+gzZ/wDfp69soro+qUux4/8ArJmX86+5f5Hif/CjdW/6DNn/AN+npk/wS1W3t5Jm1izIjQsQIn5wM17fVbUf+QXdf9cX/wDQTSeEpW2KjxJmLkk5r7l/keR/Alt19q59Yoj+rV7LXjHwH/4/NW/64w/zavZ6eE/goz4j/wCRnU+X5IKKjnuIrW3ee5kWKKMbndzgKPUmnI6yRq8bBkYZDA5BFdR89dXsOr52ttAh8T/Fm/0q5meCOa7uWLxgEjaWPevomvDPCH/JeLn/AK+bv/2auPEpNwT7n1OQ1JUqWJqQdmoXX4nR/wDCjNK/6C97/wB8J/hR/wAKM0r/AKC97/3wn+FekXd9BYorXLMN7bVCoWLH0AAJplrqlneWslxbzBo4iVkyCpQjqCDyK2+qU7c3Loed/rBmHNy+21+X+R51/wAKM0r/AKC97/3wn+FH/CjNK/6C97/3wn+FegDWrRbG3upvMiW4/wBUhQs7d+FGT05psuvWMenm9V2khWURSEKQYyTj5gcEYzVfUY/yE/6x461/b+fT/I88uPgjpcNrLKNWvCUQsAUTnA+lZfwL51jVj/07x/8AoRr2G9IOm3BHIMTY/KvHfgV/yGNV/wCveP8A9CNckqcKdaHKu57dDHYjGZXinXlzW5bfee1UUUV3nx4UVUvdUsdNeBb+5jtzcNsjMhwGbGcZq3nPSldCUk3ZBXzponhuDxX8SNQ0y5nkgjMtxJvjAJyr9OfrX0XXh3w5/wCSy3//AG9f+hiuTEpSlBPufU5FVnSw+KqQdmo3X4nQ/wDCjNK/6C97/wB8J/hR/wAKM0r/AKC97/3wn+Fek3V7bWMQkvLiOBCcBpGCjP40sV5bT2v2mCeOSDBPmI2V4681t9Vp2vynn/2/mF+X22vy/wAjzX/hRmlf9Be9/wC+E/wo/wCFGaV/0F73/vhP8K9GfU7OKyS7muY4oHxteQ7Qc/WhtTsxbR3H2hGhkYIsincpJOByPen9Th/KL/WHH/8AP7z6f5HmGofBbTLLS7q6TVbxmgheQKVTBIUnHT2rK+BZz4i1M+tmv/odev67/wAi7qP/AF6S/wDoBrx/4Ff8jBqX/Xmn/odckqcYVocqPdw+NxGLynFOvLmta33nt9FFFegfFhRRRQBleKP+RR1f/rym/wDQDXjHwh8P6V4h1HUotZso7tIYI2jVyflJJz0PtXs/ij/kUdX/AOvKb/0A15V8CP8AkL6v/wBe8X/oTVxVknXgmfV5ZOdPKMVKDs046r1R6F/wrbwh/wBAK3/Nv8aP+FbeEP8AoBW/5t/jW2usWD6qdNS5RrsKWMS8kD37U9tQiGqrYBXaUxeazAfKq5xya7Pq6/l89j53+08T/wA/pb2+J79tzB/4Vt4Q/wCgFb/m3+NH/CtvCH/QCt/zb/GtiLXNPmuFhjnJZ2KIxjYKzDqAxGCfoalstRjvpLmNEdHtpPLdXGDnGQfoc0PDpbw/AFmmJe1aX/gT/wAzg/HPgXw1pfgnU73T9IgguYYgY5FLZU7gPWmfBD/kU77/AK/T/wCgLXR/Ej/knWsf9cR/6Etc58EP+RTvv+v0/wDoC1ycqjiFZdD6CNarWySpKpJyfOt3fselUUUV2HywUUUUAFeFfF6EXHxHtIWJAkt4kJHbLsK91rw74rf8lQ0//rlB/wCjDXJi/wCH8z6XhptY5tfys6H/AIUZpX/QXvf++E/wo/4UZpX/AEF73/vhP8K9Qoq/q1L+U5P7dzL/AJ+v8P8AI8v/AOFGaV/0F73/AL4T/Cj/AIUZpX/QXvf++E/wr1AEHoc0EgDJOBR9Wpfyi/t3Mv8An6/w/wAjy/8A4UZpX/QXvf8AvhP8K4iLQ4vDnxisNKt5Xmjt72HDuACcgN2+tfQ9eGa9/wAnAw/9f1v/AOgLXPXowgouK6o9vJ8yxeLlWhXnzLkk+nke50UUV6B8WFFVINVsbm+nsoLmN7m3OJYc/MvAPT0561bpJp7CUk9jjPizx8OL7/fi/wDQxXlnhT4Z3/izRf7StdRt7ePzGj2SIxORjnj616l8Wf8AknF9/vxf+hiqvwa/5EEf9fUn9K4qlONTEWl2PsMDjK2CyaVWi7Pnt36I5H/hRurf9Bmz/wC/T0f8KN1b/oM2f/fp69sorT6pS7HF/rJmX86+5f5Hif8Awo3Vv+gzZ/8Afp6P+FG6t/0GbP8A79PXtlFH1Sl2D/WTMv519y/yPnDxj8P73wZZ21xeX0FytxKY1ESMpUgZzz9K9j+GBz8NtIz/AM83/wDQ2rmfjp/yA9J/6+2/9ANdN8MP+SbaR/1zf/0NqyowjDEOMe3+R6GZYqri8mpVqzvJz/8AkjrKKKK7z40KKq2epWWoGUWVzHM0LbJFRslD6EdulWqSaewk1JXRieM7y40/wVq13ZStDcQ2zPHIvVSB1rxjRtW+I/iCCSbR9Qv7qOJ9jsrRjBxnHOO1eweP/wDknut/9ej/AMq5H4Gf8i9qn/X4P/QBXHWTnWUbtaH1mW1YYbK6uI9nGUlJL3lfexz32X4t/wDPXUf+/kVH2X4t/wDPXUf+/kVe40VX1ZfzP7zD+35f9A9P/wAB/wCCeHfZfi3/AM9dR/7+RUfZfi3/AM9dR/7+RV7jRR9WX8z+8P7fl/0D0/8AwH/gngGrX/xL0Ky+16tfX9tBvCb2eM/Meg4r034W6tf6z4JS61W6e6uPtEieZJjJAPA4qj8aP+RBH/X5F/Wn/Br/AJJ8n/X1L/OopxcK/LdtWOzG1oYvJ/rHs4xlz291W6He0UUV3Hx4UUUUAFct8QPFV14Q8PxX9lbwzySXCxFZicAFWOePpXU15z8bP+RLtf8Ar+T/ANAesq0nGm2j0srpQrY2nTqK6b1OUPx01kHB0zTgf95/8aP+F66z/wBA3Tv++n/xrufhlptjP8PNNkns7eR2D5Z4lJPzt3IrrP7H03/oHWn/AH4X/CuaNOtKKlz7nu4jG5VQrTpPC35W18XY8a/4XrrP/QN07/vp/wDGj/heus/9A3Tv++n/AMa9l/sfTf8AoHWn/fhf8KP7H03/AKB1p/34X/Cq9jW/n/Aw/tPKv+gT/wAmPGv+F66z/wBA3Tv++n/xoHxz1o9NM08/Rn/xr2X+x9N/6B1p/wB+F/wryL43Wlta3WkC1t4oQyS7vLQLnlfSs6sa1ODlzndl9bK8diY4dYW1763fRXPYdPuWvNMtbl1CtNCkhA6AlQf61Yqjof8AyL2nf9esX/oAq9Xetj42okptLuFFFFMzCiiigDK8Ta7H4a8PXOrTQNOluFzGhALZYDqfrXn/APwvWx/6Ad1/3+Wum+Kn/JNdU+kf/oxa5D4PaBpOr+Hb+XVNOtruRLvarTRBiBsU45+tcdWdT2qhB20Pp8vw2BWXTxeKg5WlbRtdF5+ZZ/4XrY/9AO6/7/LR/wAL1sf+gHdf9/lruf8AhCvDP/QB0/8A8B1/wo/4Qrwz/wBAHT//AAHX/CnyYj+ZEfWsl/6B5f8AgT/zOG/4XrY/9AO6/wC/y0f8L1sf+gHdf9/lruf+EK8M/wDQB0//AMB1/wAKP+EK8M/9AHT/APwHX/CjkxH8yD61kv8A0Dy/8Cf+Zw3/AAvWx/6Ad1/3+Wu/8Ma/H4m8P2+qwwPbpMWAjdgSMMR1H0rgfiz4c0bSvBiXGm6Za2s32uNfMhiCnBDZGRXR/Cn/AJJvp3+9L/6MalTlUVXkm76FY7D4GWXRxeFg4tytq2+j8zsaKKK7D5kKKKKACiiigAooooAK8m+LRI8Q2WCR/ovY/wC21es1y3inwPB4ov4bqa+ltjFF5YVEBzyTnn61hXg5wtE4Mwozr4dwprXQ8Syf7x/OjJ/vH869R/4VDaf9Be4/79LR/wAKhtP+gvcf9+lrg+rVex81/ZWL/l/FHltPhmlt5lmt5HikQ5V0Ygg+xFXb7SZYPEFzpdmsl08M5iTavzNg4z7V2WgfC26mkjn16VYYcBvIjOXPsT2/Cs405ydkjlo4WvVnywW34fMZ4Y+IuspJ9lvbSbV/Qwr+9H5DB/SvV0bfGrEFdwBweoqjpWiadoluYdMtUgUnLEcs31J5q/Xq0oTirSdz7LCUa1KHLVnzP+uu7PDfA/8AyXK8/wCut3/M17JqNpcXIBh1CSzjRSWMSrkn3LA8fhXjfgf/AJLlef8AXW7/AJmvW9etdQvY4razAFsx/wBIIcKzL/dH19anAettXufWcTu1aGjfuR2+Zk/2xfTeC45GmzdXE/2VJ1G3cC+0OPw5q34rn1TT9D8/TbiOFIQvmOw3O3zAYGePxNJ4gjlh8OxSpaeSLGeOXykYNhFIzj8Ks+IrW61bQ5LSxiR/PVSJGk2gYYH+leunDnjKytzPt5Hxcoz9nOF3flVt99fmJrU00Gl2WoJM6eRJG8oU4DqcAgj8c1o6gd2k3JHQwOf/AB01ka4zvoVtp88flzXUscAUNu6EEn8hWvfrt0i5X0gYf+OmuWqv3a9X92n/AATtw7ft/lH79f8AgHkXwH/4/NW/64w/zatzxX8QNY0XxNd6fZpamGHbtMkZJ5UH196w/gP/AMfmrf8AXGH+bV0fif4dajrniO61G2vLaOObbhZA2RhQOw9q8inz/V48m563GixDx0/q973jt25TltS+ImtarplxY3KWohuEKOUjIOD6c1F4Z8b6j4ekWJna6suAYHb7o/2T2+nSrmq/DXUtI0m5v5721eO3jLsqBskD04rN8NeD9R8SSloF8i1UjfPICBj/AGfU1k/bKavufnb+vKvG9+fp6HsWg+JNO8Q2iy2My+Zty8DEb4/qP615F4Q/5Lxc/wDXzd/+zV6z4e8Kab4bh/0KLdOyhZJ35Z/8PoK8m8If8l4uf+vm7/8AZq6KvNenzb3P1Th/2v1TFe1tzeze3zPa768t7C1a4umVVXpnqT6D3rmLyG7g8Have3KeTPeZlMaj/VqQAAffA5rotS0iy1eGOPUIfNSN96AOVw3rwRUSaBYRWNzaxJII7lCkgeZ3yP8AgROK9ilUpwS73X3fefK1adSculrPr1t6GeP3fizTEPEAsGEYxgbsr+uKp36pJa+J9uPI2dR03hOfx6VtLosV1pFrZ6sBO9uBh0ZlOQMAgjB6Uy+0JJNDOmaftt4XceZkkkrn5uepJHGTWkasFJa+Xlve5i6NRwbt526/Da39Mlgd5PCyPJne1nls+uyvJ/gV/wAhjVf+veP/ANCNew3aCPS50UYCwsB/3zXj3wK/5DGq/wDXvH/6Ea8iu74iD9T7LK4uOUYqL6KH5nsGqXL2ekXlzFgyQwPIu7pkKSK8m/4Wp4g/552X/fpv/iq9Z1S2e80e8toceZNA8a5OBkqQK8k/4Vb4h/6dP+/x/wAKnEe0uuQ+EzL63zR+r3t1sZXiHxdqPiaCCLUVgCwuXXykI5Ix3Jra8K/EW70opa6uWu7XKqshPzQqOPT5u3vWJr/hLUvDUEMupeTtmcovlvu5Az6Vq+Fvh9f6y8V1fr9lsdysQ/DyoRn5cdPxrkj7X2mm54lJ436z7t+fr/wT1+yvrXUbYXFjPHPE3R42BFeK/Dn/AJLLf/8Ab1/6GK9m0zS7PR7JbTToFghUk7RzknuSeteM/Dn/AJLLf/8Ab1/6GK6q1+anfufq+S831LFc2/J/mez37WkEH2q+WPbACwZwOOO3vXK3Xn2vgbULgRG0+2SmRYuhjViB+BOM/jXTanpNvqqwrdNIBC/mKEbHzdsjvUd9pAvNHubJ7iWTzkwGkYHae3avXo1IQSv3V/RHyVanObdl0dvVozJVV/HVlbS4MMNiXiQ9NxJBP5AVm3KBNO8WQxcQwuJIwOivsDHH4810zaUl2lpPeBo7y3XAlhfBGRgjPcGq99oSnRZtPsFC/apAZ5HbJIJ+Yk9zjNaQrRTSv2X3O9znnQm4tpd35/Dawt3M1x4JuJpPvSae7H6mM15R8Cv+Rg1L/rzT/wBDr17WI1i8MX8aDCrZyAD/AIAa8h+BX/Iwal/15p/6HXjV2niINeZ9xlSccmxSe/unt9FFFdR80FFFFAGV4o/5FHV/+vKb/wBANeVfAj/kL6v/ANe8X/oTV6r4o/5FHV/+vKb/ANANeVfAj/kL6v8A9e8X/oTVx1f94h8z6jL/APkTYr1j+aPR5baG38d2At4kjD207tsXGSSuSfenSeY11r7oP36wBIyBzjaSMfjVqXwvpM14bqSCUzZJ3/aJBjPXHzcD2qYadLH4gN9FIohkgEUsZzkkEkEfnXse0g0rPZdfW58T7KabdlZyvp6W7L1MC/IfwRpAt+Z/Mt/L29d2Rn8eua1bNmXxhexr9xrWN2/3skfyq1baDptpem7t7fbKSW++xUE9SFJwPwFLpthNbXd7c3UiPJcygrsH3UAwB/P86cqsHFpef4tflYmOHqKUW/JaeSf53Mj4kf8AJOtY/wCuI/8AQlrnPgh/yKd9/wBfp/8AQFro/iR/yTrWP+uI/wDQlrnPgh/yKd9/1+n/ANAWvHl/vK9D7Sj/AMiOr/jX5I9KooorrPmwooooAK8O+K3/ACVDT/8ArlB/6MNe414d8Vv+Soaf/wBcoP8A0Ya5MX/D+Z9Jw3/vr/ws9ru45pbSRLWYQSsMLIV3bffFYPhieaHStR+1XM1yba7lUPK25iBW/c+f9nf7GIzNj5PNJC598c1iaNpGp2a3kN+bN4LqSSU+UW3Bm7cjpXqU2vZSTa6ep8jUT9tBpPr6FGB5ItE0m3WSSOTVp980qEqw3AucHtwMVbktNmqXekmaaS2vLQsqySFzGRwcE5POf0qKxsbjUPDlpGmINQ0yXbGZAdu5Pl59QR/OrZgvLJL3V9UMctwkBWOK3BIUDnAzySTXROSu0nrr999P0OSnFuKbXbXy+0n+P3k/he7e88N2ksxLSBSjE9ypI/pXkWvf8nAw/wDX9b/+gLXr/huyew8PWsEoxJtLOPQsSf615Br3/JwMP/X9b/8AoC15GYW5/d25j7PhW9p82/sn+h6N4+8TX3hqzspNOWFmnkZW81SeAM9iK4n/AIWp4g/552X/AH6b/wCKrtfH/hq/8SWdlHpvlboJGZ/MfbwRj0riP+FW+If+nT/v8f8ACuet7bnfLex+fY7697d+xvy6bHPza/qEmvyazFN9nvZG3F4hgDjHQ57V6Z4S+IltqgWz1gpa3KooWZ3wsx6H6HPavMbjRL+31yTSFh8+9RtvlxHOTjPH4V6N4T+G0Vpi78QJHcSna0UIJxGevzep/SsaHtef3fmcmXvGe2fJtfW+3/Dl34snPw3viOm+L/0MVV+DX/Igj/r6k/pVr4sgD4b3wAwA8X/oYqr8Gv8AkQR/19Sf0rp/5ifkfqUf+RC/+vn6G74vutUstPW4064jgiSRBIdu52ywGBngDmrGsXUm/TLON2RrycBmVsEKo3H88Y/Gk8TWV5qemNZ2cKtuZH3tJtA2sDjGPaodWWWO40a9mj8sW9xslAbO0MpXOfqRXsU+VxitL6/lp+J8VV51Um9bWXpu72+Qwz3+salqMdpevZx2JEcYjVTvfGSWyDx04GO9RLq07zaFfmRglyxtp4wflLEcHH1H61aTTtQ0/UdSmsVimjvsOu59pjfGDnjkdKp3Nh5Fx4f0lDueKU3EjDp8o5P4k1pHkbsrWt/7a7/iZzVVJt3vd/8ApS5bfL/gnL/HT/kB6T/19t/6Aa6b4Yf8k20j/rm//obVzPx0/wCQHpP/AF9t/wCgGum+GH/JNtI/65v/AOhtXhR/3mXp/kfe4j/kQUf8b/8AbjM8c+NtU8O67FZ6eluY2gEhMqEnJYj1HpXN/wDC1PEH/POy/wC/Tf8AxVdH468F6r4h16K80/yPKW3EZ8yQqchifT3rmv8AhVviH/p0/wC/x/wrOp7bnfLex+YYr+0Pby9nzct9DAsvEOpadq0uoWNwYZppC8qqPlfJzgj0r1nwt48sfEC+Tc7LO8BAETvxJ/unv9K8jtND1C+1d9Ns4DNcRuUfb91cHBJPpXqvhT4e2eihLrUgl3fq25WGdkfpgdz7mpw/tebTYjK3i+f3fh632/4f+maHj/8A5J7rf/Xo/wDKuR+Bn/Ivap/1+D/0AV13j/8A5J7rf/Xo/wDKuR+Bn/Ivap/1+D/0AVvL/eI+h+nYf/kSVv8AEv0O3vNdlttct7BbCQxSuIzcsdq7iCcAd+BU11qUia/Z6dAoPmK0szEZ2oOn61l+I9UtItW02N5G329yHlAjY7RsPOQPcVP50f8AwnUTZ4uNP/dHkZw2f5V7CprlUuXo/wCv1PinWlzyjzdY/K/T9BzeIbj7Jc36WKmxtpCjOZPnZVOCwXGMfjVqDUpX102shQ281uJrdh1POGB9eoNc/wDatnh++0CSKVdQdpIY0KHDhicMG6Ywc1oi1aPxHo1upBa0tJPMI9PlA/WrlSgrq1t/uSumZxrVXZ3vtfyblZr7vyML40f8iCP+vyL+tP8Ag1/yT5P+vqX+dM+NH/Igj/r8i/rT/g1/yT5P+vqX+deH/wAxXyPvX/yIP+4n6He0UUV2Hy4UUUUAFec/Gz/kS7X/AK/k/wDQHr0avOfjZ/yJdr/1/J/6A9YYj+FI9bJf+RhS9TX+Fn/JN9M+j/8AobVr32q31gBNNZRfZzOIv9d85BONwGMfhnNZHws/5Jvpn0f/ANDamau+mXNxI1glz/bKzYiRg5CuD1wflx7114KCnGKa0sv6/wCCedndRwxVZxdnzS/N+WvodBJqTHXUsotoRIDPOSM4GcKB6d/yqkniKaS0jv1sl+wPN5Yk8z5wu7G/GMY/GoreN/8AhLtQimx5lxYRlT9CwP6ms21mP/CMw+HWikGoK4hdNhwoDffz0xjmuyNKFlpfb7ne7+R4861RN62+L71ay+e50VnqcsuvXun3AX92qywsv8SH/wCvmvMPjt/x9aN/uTfzWvQrdd3jx9nIg09Uc+5bIrz347f8fWjf7k381rzMwilS07I+r4VlKWZR5ujl+T/4Y9V0P/kXtO/69Yv/AEAVeqjof/Ivad/16xf+gCr1VHZHmVv4kvVhRRRTMgooooA5D4qf8k11T6R/+jFrB+Bv/Ir6l/1+/wDtNa3vip/yTXVPpH/6MWsH4G/8ivqX/X7/AO01rjl/vK9D6ej/AMiCr/jX/tp2+o61LYPKV0y5nggXdLMpACjvgE5bA9KkuNYVVsDaR+f9ucBMnGFxkt/n1qnr01ldWN9C2pNDJFEytEGxk4yBjqc+1VtMlmkuNDkvIhAzWrqIwMYbjt9K9mNOLgpNf1a/9WPiZ1Zqo4qWny01S/XqXrzxCbe4nW3sJ7uK2/180RXCHqRgnJIHpU39tRnVrS1CZivITJDMDwSOduPpzWRp15BpdprMF9IEmW5lk2H7zq3KkDvUIt5LZvClvIMTo5LD0HlnNV7GG1v+Dpe5m69TdP8A4HvJW+a/IzvjT/yIaf8AX7H/ACatD4U/8k307/el/wDRjVn/ABp/5ENP+v2P+TVofCn/AJJvp3+9L/6MavFX+8v0Pt5/8iGP/Xz9GdjRRRXWfNhRRRQAUUUUAFFFFABRRRQAUUUUAV7extbSSWS2t44nmcySMq4LMepNWKKKNhJJaIKKKKBnhvgf/kuV5/11u/5mvcq+Y9Q1a+0Tx3ql9pc5t7lbudVkABwC5z1q3/ws3xf/ANBl/wDv0n/xNeXRxEaScWup+hZlkeIzCcK1OSS5Utb/AOR9IkAjBGRR06V83f8ACzfF/wD0GX/79J/8TR/ws3xf/wBBl/8Av0n/AMTW312n2Z5f+qeM/nj+P+R9IFQSCQCR0OOlV9R/5Bd1/wBcX/8AQTXzv/ws3xf/ANBl/wDv0n/xNNk+JPi2WNo5NYcq4KsPKTkH8KHjYW2ZUeFMYpJ88fx/yOs+A/8Ax+at/wBcYf5tXs9eNfAoYv8AWAOgii/m1ey1phP4KPP4j/5GdT5fkiK5toby2kt7qJZYZF2ujDIYehp8caQxrHEoRFGFUDAAp1FdR89ZXuFeGeEP+S8XP/Xzd/8As1e518veIbq4s/G2rT2k8tvKt7MBJE5Rhlz3FcWKly8suzPreHKLxCxFFO3NG33n1DRXyr/wkmuf9BvUf/AuT/Gj/hJNc/6Deo/+Bcn+NR9ej2Or/VGr/wA/V9zPqqivlX/hJNc/6Deo/wDgXJ/jR/wkmuf9BvUf/AuT/Gj69HsH+qNX/n6vuZ9R33/IOuf+uTfyNeOfAr/kMar/ANe8f/oRrz8+ItbZSG1rUSCMEG7fn9a9B+Bn/Ia1b/r3T/0I1CrKtWjZbHRUymeW5ZiFKXNzcv4P/gntNFFFemfAlHUdGsdWktn1CBZ/szl41b7obGMkd6vAYGBwKKKVknclRSbaW4V4d8Of+Sy3/wD29f8AoYr3GvljVLu5svFOpTWVzNbS/aph5kMhRsbzxkVx4qXK4S7M+u4doPE08RRTtzRS/M+p6K+Vf+Ek1z/oN6j/AOBcn+NH/CSa5/0G9R/8C5P8aj69HsdX+qNX/n6vuZ9VUV8q/wDCSa5/0G9R/wDAuT/Gj/hJNc/6Deo/+Bcn+NH16PYP9Uav/P1fcz6b13/kXdR/69Zf/QDXj/wK/wCRg1L/AK80/wDQ64N/EOtSIySazqDowwytdOQR6Hmu++Bv/Ix6p/16L/6HUKsqtaLSOqeVSy3K8RGUubmt+DPbaKKK9M/PwooooAyvFH/Io6v/ANeU3/oBrx/4MarYaVqmqPqV5BaLJBEEM0gUMQWzjNeweKP+RR1f/rym/wDQDXzLpWi6nrbPHpNjNePEoZ1iXO0HoTXn4mTjVjJK59rkNCniMvxFKpLli2te33n0t/wl/hz/AKDmn/8AgQv+NH/CX+HP+g5p/wD4EL/jXz5/wgfir/oX73/v2P8AGj/hA/FX/Qv3v/fsf40vrVX+T8zX/V7Lv+gn8Y/5n0H/AMJf4c/6Dmn/APgQv+NH/CX+HP8AoOaf/wCBC/418+f8IH4q/wChfvf+/Y/xo/4QPxV/0L97/wB+x/jR9aq/yfmH+r2Xf9BP4x/zPYfH3iXRL3wHqtvaatZzzSRAJHHOpZjuHQZqh8EP+RTvv+v0/wDoC15NeeEPEOn2cl1faNd29vEMvK6YCj3r1n4If8inff8AX6f/AEBaVOpKpXTkraDx+CoYPKJwoz505J308ux6VRRRXpHwgUUUUAFeHfFb/kqGn/8AXKD/ANGGvca8F+MzFfHsTKSrCzjIIOCDuauTF/w/mfS8Mq+Ot/dZ71RXyr/wkmuf9BvUf/AuT/Gj/hJNc/6Deo/+Bcn+NZfXo9j0f9Uav/P1fcz6qor5V/4STXP+g3qP/gXJ/jR/wkmuf9BvUf8AwLk/xo+vR7B/qjV/5+r7mfVVeGa9/wAnAw/9f1v/AOgLXEf8JJrn/Qb1H/wLk/xq94UuZ7v4gaPNdTSTytexbpJXLM3PcmsqmJVW0Uup6GByOeXKrVlNO8JLb5/ofTlFFFeqfnJRtdGsLTUrm/ht1F1ctmSU8seAMZ7DjpV6iikklsTGKirJHF/Fn/knF9/vxf8AoYqr8Gv+RBH/AF9Sf0q18Wf+ScX3+/F/6GK8U0jxn4g0Kx+x6TqLW9vuL7Aink9TyK4atRUq/M+x9rluAqY/J5UabSfPfX0R9P0hAYYYAj0NfN//AAs3xf8A9Bl/+/Sf/E0f8LN8X/8AQZf/AL9J/wDE0/rtPszL/VPGfzx/H/I+kaTA3ZwM9M183/8ACzfF/wD0GX/79J/8TR/ws3xf/wBBl/8Av0n/AMTR9dp9mH+qeM/nj+P+R3/x0/5Aek/9fbf+gGum+GH/ACTbSP8Arm//AKG1eD614r1rxFDFFrN811HC29AUUbTjGeAO1e8fDD/km2kf9c3/APQ2qaFRVK7ku3+Rvm2Dngsnp0KjTan08+Y6yiiivQPiinYaTY6YZmsbdInncvK4HLnJOSe/WrlFFJJLRCjFRVkjnfH/APyT3W/+vR/5VyPwM/5F7VP+vwf+gCuu8f8A/JPdb/69H/lXi3gv4h3HgywubW306K7FxL5pZ5Sm3gDHAPpXFWnGFeMpdj6/LcLVxeUVqVFXk5Ly7dz6LqvPYwXN1b3Eqky2zFo2BxjIwa8g/wCF63//AEAbf/wJb/4mj/het/8A9AG3/wDAlv8A4mtVjKS1TOF8NZk1Z0198f8AM9nqtFYQRahNeqGM8yqjEtnAHQD0615D/wAL1v8A/oA2/wD4Et/8TR/wvW//AOgDb/8AgS3/AMTQsZSWzB8NZk96a++P+Z0/xo/5EEf9fkX9af8ABr/knyf9fUv86848X/E268XaGNNn0yG1XzVl8xJix4zxggetej/Br/knyf8AX1L/ADrGnUjUxF49j08Zg62DyT2VZWfPfv08jvaKKK7z4wKKKKACvOfjZ/yJdr/1/J/6A9ejV5z8bP8AkS7X/r+T/wBAesMR/Cketkv/ACMKXqa/ws/5Jvpn0f8A9DauvryHwR8TdA8P+D7LTNQ+1/aIA2/y4dy8sT1z71v/APC5/Cv/AE/f+A3/ANeop1qagk30OrH5XjqmLqzjSk05O2nmdxLZQy3kN0wPmwhgpBxweoPr0qeuA/4XP4V/6fv/AAG/+vR/wufwr/0/f+A3/wBetXiKb3kcSybHptqjLXyO2t7CC1ubm4iU+bcsGkYnOSBgfhgV5L8dv+PrRv8Acm/mtdJ/wufwr/0/f+A3/wBevPfid4x0vxdNp76T5+LZZA/nR7OuMY/KufFVoTptKV3oe1kOWYvDY+E6lJxjrrbun+p7nof/ACL2nf8AXrF/6AKvVR0P/kXtO/69Yv8A0AVersjsj5et/El6sKKKKZkFFFFAHIfFT/kmuqfSP/0YtYPwN/5FfUv+v3/2mtb3xU/5Jrqn0j/9GLXF/CTxTomg+H76DWNSgtJZLveiyHBK7FGfzBrim0sSm+x9Xhac6uRVI002+fpr/KewNFGzbmjUt6learX2nRX8ltI7uj20vmxsnrgjH0waxv8AhYvhH/oP2n/fR/wo/wCFi+Ef+g/af99H/CutVoxd1I+ell2JkrOlL/wF/wCR0bRRswZkUsOhI5qq+mwyavHqLljLFGY0X+Fcnk/Wsb/hYvhH/oP2n/fR/wAKP+Fi+Ef+g/af99H/AAoVaK2l+ISy7Ey3pS/8Bf8AkYfxp/5ENP8Ar9j/AJNWh8Kf+Sb6d/vS/wDoxq5b4qeLdB1zwelrpOqQXc4ukcxxkk7QDk/rXU/Cn/km+nf70v8A6MauWLTxLa7HvV6VSlkcY1ItP2nXTozsaKKK7D5cKKKKACiiigAooooAKKKKACiiigAooooAKKKKAPA/DOl2WsfGS+s9Uto7q3ae6YxyDIJDHFer/wDCu/CP/QAs/wDvivMvA/8AyXK8/wCut3/M17izBFLMcKoySe1cWGhGUXddT6vPsTXpYinGnNpckdm13Ob/AOFd+Ef+gBZ/98Uf8K78I/8AQAs/++K17LWbHULKS7tZ98ERIdypGMDJ6/WlutXsbLTRf3c4it2UMGYHJz04612fV1fl5NfQ+c/tKu1ze2dv8T/zMf8A4V34R/6AFn/3xUF98PvCcen3Dx6DZqyxMVITocGuibUrVdNW/aTFsyqwfaeh6H9aXUCDpN0RyDA+P++TUyoxS1j+BpTzDEymrVZdPtP/ADPIvgQc3mrk/wDPGL+bV7NXjHwH/wCPzVv+uMP82r2esMJ/BR6nEf8AyM6ny/JBRRRXUfPhXhXhONJfjrdJKiupubvKsMj+Kvda8M8If8l4uf8Ar5u//Zq5MR8UPU+lyT+Biv8AA/1Pa/sFn/z6Qf8AfsUj2VjGhd7a3VVGSTGoAFZ+v3bkRabayrHPdffYtgxxD7zf0/GuctH/AOKCS1jc/vr37OcNkhWkwf0r1KeG54qT6tL/AIJ8jVxrhNx1dk3v26HZrZWLoHS2t2VhkERrgimta6ekiI8FsryZ2KUXLY9KxVhj1Hxfc2d2gktrO2QRwsPly38WPXtWU88lt4XuX3tI2l6mFhdjltocDGfoxFVHCqTSvvb8diJ46cYuXRX69t/1OrvrCzGn3BFrBnym/wCWY9K8k+BX/IY1X/r3j/8AQjXsN227S529YWP/AI7Xj3wK/wCQxqv/AF7x/wDoRryaqtWh8z7HL5OWVYu7/l/M9qooorrPmgooooAK8L+HsUc3xiv0mRZF/wBKOGGR98V7pXh3w5/5LLf/APb1/wChiuTEfHD1PpMn0wuL/wAH+Z7T9gs/+fSD/v2KZLbadBHvngto1zjcyKBmsLxNdLqH2jTYp9iW0LTXBDYJOPkT8ep+nvSxbb228O2znehVZm5zkonQ/jXqRw65VKX9K1z5KWMlzuEdfn1ul+pv/YLP/n0g/wC/Ypn2XThMIfItvNK7gmxc49cVjeIrEDUrC+M8xb7THGsW/CKOcnHqfepdeP2bW9Fu0++bg27e6sp/qBUxoxlaz3v+A5YupFyv0t16NlnXLG0Xw/qJW1hBFrKQRGOPkNeTfAr/AJGDUv8ArzT/ANDr2DXf+Rd1H/r1l/8AQDXj/wACv+Rg1L/rzT/0OvMq/wAeB9jlzbynF3/u/me30VwnxI8Q6poT6eNKufI84Sb/AJA2cbcdR7muI/4WD4m/6CX/AJCT/CrniIwlys+BxGaUaFR05J3Xp/mev2PiHTdR1O6063n/ANLtWKyRMMHg4yPUVp185PqV4+pSagLmRLuRi7TRnY2T16dK9K8JfElLoiy8QMkMvyrFOqnD9vm9D3z0qKeJUnaWhjhM2hVlyVNH0Ou8Uf8AIo6v/wBeU3/oBryr4Ef8hfV/+veL/wBCavVfFH/Io6v/ANeU3/oBryr4Ef8AIX1f/r3i/wDQmoq/7xD5n6Hl/wDyJsV6x/NHst1dQ2drJcXLiOKNdzMewqq2tWS6Oupq7SWz42sqnLZOBx9aztclafUobS4tbhrKJfOd44WcO4+6vA6dz+FZumSrceG9EhKOI/tOH3xkA4LEdevNexCgnBSff8Nfx0PiamIkqjhHs/vVvw1OkvtZsdMWE386wGdgsat95ifYVJcajDbXlrbS7g10SsbAfLkDOCfoKxfGsER0dJzEpmFxCofb8wHmDjNWfEpMenW0sf8ArUuYtn4sAf0qY0oSUH3bX5W/MJ15wlUT2ik/zv8AkU/iR/yTrWP+uI/9CWuc+CH/ACKd9/1+n/0Ba6P4kf8AJOtY/wCuI/8AQlrnPgh/yKd9/wBfp/8AQFrypf7yvQ+yo/8AIjq/41+SO91XV7LRbL7XqU3kwbgu7aTyenSsX/hYvhj/AKCX/kF/8KqfFL/kSz/18x/1rxrI9amtXlTnZH57mGZVcNW9nBK1uv8Aw56XrXxK+zeIoJdFnW708xqJ42Qjnc2cZ6HBH6V3Gh+I9N8Q25l06fcy/fibh1+o/rXz5Wx4YstZvNZj/wCEfLx3CHJmBwqD1Y+ntWNPET5tdbnDhs0r+195cyfRfoe/14b8Vxn4oWAPIMMGf+/hr2yzW4SyhW9kSS4CASOi7QT7CvFPit/yVDT/APrlB/6MNdGK/h/NH6fw0741v+6z2n7BZ/8APpB/37FH2Cz/AOfSD/v2Ki1bUotK097iUjd92Nc/fc9BXPaIDp2rar5lx9ol+zxyu27O58MTivRhQ5oOfb8T5epinCpGF9/Pbf8AyOjjttOm3eVBbPsba21FOD6Uslpp8MbSS29siKMszIoArL0nThc+EIITNLC1yglkkibaxLfMefxxTPD9pHP4Wns3y0TPLF8xycZI6mnKjBXd9nYiOJqycV/Mr7/gbIsbJlBW1tyDyCI1rxLWo0i+P0CRoqKL63wqjAHyLXrnhOd7jwxaNISzKGQk99rEf0ryXXv+TgYf+v63/wDQFrgxsORqPaR9Vw1VdV1ZvrTb/I9zoooroPnQooooA4v4s/8AJOL7/fi/9DFc78LvCOga14NF3qulW91cfaJF8yRcnAxgV0XxZ/5Jxff78X/oYqr8GzjwACen2qX+lccoqWJs+x9RRq1KWRuVOTT9p006I2f+Fd+Ef+gBZ/8AfFH/AArvwj/0ALP/AL4qAXs9/wCJdPvYp5ks5JXhiiDEJKqqTvI75P8AKpJNSeGTxBqgG5rJRDED0G1dx/Vv0r0vqUdrK9u3nax8x/a+J39pK1/5n2vcf/wrvwj/ANACz/74o/4V34R/6AFn/wB8VWlW40m20rUVvLiWW4mjjuVkkLK+8jOFPC4z2rQ8PO0Oqavp5YskFwJI8noHG7H55pSwkFFyVnby87fqKGb4qUlFzkr/AN59U2vyZ5t8XfDOi6DpGmy6Pp0Fm8tyyu0S4LDYTiu7+GH/ACTbSP8Arm//AKG1cz8dP+QHpP8A19t/6Aa6b4Yf8k20j/rm/wD6G1ebTSWJkl2/yPq8XUnUyKlKbbfO99f5jrKKKK7T5QKKKKAOd8f/APJPdb/69H/lXkvw7+H9h4x0u8ub66uYGt5xGoh24I2g85B9a9a8f/8AJPdb/wCvR/5VyPwM/wCRe1T/AK/B/wCgCuKrFTrxUux9Xl+Iq4bJ61Sk7PmWv3En/Cj9E/6CV/8Amn/xNH/Cj9E/6CV/+af/ABNdzrd9NbwJb2G03lwdsef4R3fHoBWUt/eXng+yDXDC7vJBH5q/Kfv8njpwK7Y4CnKKlZau3/BPEnxHj4ScfattJvp06eupzf8Awo/RP+glf/mn/wATR/wo/RP+glf/AJp/8TXXeI59TsbKOSwmjigiKCRmG53ywGOeBwevNO8QXE1kmn30crKsdwiSqDw6v8pyPbOaUcDTlayWopcR5hHmTqP3Un02/pHlPj74bab4T8NDUbK8uppPPSLbLtxg59APSu1+DX/JPk/6+pf50z40f8iCP+vyL+tP+DX/ACT5P+vqX+dcUIRhibR7H0OJxNbE5H7StK75/wBDvaKz9X1yw0K2SfVJvJjd9inYWycZ7D2rI/4WL4Z/6CB/78P/AIV1ucU7NnxE8RRpvlnJJ+p09FeYePPGOm6ppFsmiX0huIrlZMqjoQADzkgd8VJ4X+J2dlp4jwuAcXijr7MB/MVl9Yhzcpyf2lh1W9k38+h6XXnPxs/5Eu1/6/k/9AevQ4po54llgdZI2GVZTkEV558bP+RLtf8Ar+T/ANAenX/hSPqMl/5GFH1PNdG+G3iLX9Jh1LTo7VrabOwyT7TwSOmPUVe/4U94t/542X/gT/8AWr1P4Wf8k30z6P8A+htWv4h17+xLdTFbm5nYFvLDbcIOrE+2RWVDAxqqKV7s9rH8UYvCVqsfd5YtrZ9H6niv/CnvFv8Azxsv/An/AOtR/wAKe8W/88bL/wACf/rV7fdalMl3p1vbIhe6JaTdk7EAyT+ZA/GoLvX5LbxHaaYLGQx3BI+0Mdq5AzhR3rWOWxlt679jhlxji46vl6LZ9fmeL/8ACnvFv/PGy/8AAn/61YPiXwjqvhRoE1hIVNwGMflSb+mM5496+j/t88fiIWUyp5EsHmROBg7gcEH8wa8t+O3/AB9aN/uTfzWuXEYSNKnzI9rJ+IcVjsbGhUStrfTsr9z1XQ/+Re07/r1i/wDQBV6qOh/8i9p3/XrF/wCgCrF3d29javc3kyQwxjLyOcBe3Wu9fCfD1mlUk33ZNRWL/wAJj4c/6DVl/wB/hWT4p8XaTJ4Xv00vWYDdmL915M3z5yOmKl1IpXucU8VRhFy5k7eaOworzXwv8TQfLs/EWF6Kt2o46fxj+or0aCeK6t0nt5FlikUMjqchh60U6kaivEMPiqWIjzU3/mcp8VP+Sa6p9I//AEYteGaL4S1zxFbSXGjWDXMUb+W7B1XDYBxyR2Ir3P4qf8k11T6R/wDoxawfgb/yK+pf9fv/ALTWuOtTVSuovsffZVjamByepXppNqfXzUTzz/hWXi//AKAz/wDf2P8A+Ko/4Vl4v/6Az/8Af2P/AOKr6G1LUYdLsXubknavRV6sewHuaoS+IohoMGpQwsTcsqRQucEsTjH6E1tHLVJXV97HHLjHEwdpRjtfZ7feeEf8Ky8X/wDQGf8A7+x//FUf8Ky8X/8AQGf/AL+x/wDxVe9at4is9HkjiuBJJK5XKRrnaCcAk9AM1Peai9pf2URg3Q3TlDLu+42Mjj36U/7NVk9df0F/rlibtcsbq19H1+Z836x4N1/QLEXmr6e1tAXCBy6n5j0HBPpXt3wp/wCSb6d/vS/+jGrP+NP/ACIaf9fsf8mrQ+FP/JN9O/3pf/RjVjRpqnXcV2OvMsdUx+TxrVEk+e2nozsaKKK7z4wKKKKACiiigAooooAKKKKACiiigAooooAKKKKAPDfA/wDyXK8/663f8zXq3iea4FtBbQQ3LxTyYne3iLlUHJGB69PzrynwP/yXK8/663f8zXuVc+BlyLmtfVn0HE8HOtCKdrwj+pwBvQ/hfxBHb29xCPPY/PCVCqQox9fauq1a2gk8PSs0SMYbZjGSoOz5McenFWZdKtJLK6thEFjutxlA/iJGCag/sWG70u3tNXVbkwrtyGZQeMdj6V6sq0JWa0s/0X+R8hCjOF07O6+W9/1/AzXAPwy+bp/ZoP8A45V+3eSTwerzZ8xrLLZ9dlNm8PWyaTJp+mottDMyiUEs2UB5AyeOOKvXyCPR7lFGAsDAD/gJrKtUjKDS6ts2w1KcasHLokvx/r7zyL4D/wDH5q3/AFxh/m1ez14x8B/+PzVv+uMP82r2evNwn8FH1PEn/IzqfL8kFFFFdR8+FeGeEP8AkvFz/wBfN3/7NXudeF+EWC/He5LEAfabvk/8CrkxHxQ9T6XJf93xf+B/qe03elWF+4e9s4J3UYVpIwxA/GsCfwt/Z/heW204I9wkwuFIiCl2DbgOPyrp/Oj/AOei/wDfQo86P/nov/fQr0YV5wsk9P8AI+UqYaFS7lHWzX3mRcaRNcXkep6fcfYbuSEJLvi3hl64IyORVa+0MR6TbaTbrJKs9yslxMwznB3MWPvgCug86P8A56L/AN9Cjzo/+ei/99CmsRNNeX9L7iZYWEk01vv89/vIr0bdNuAOghYfpXjvwK/5DGq/9e8f/oRr2C+lj/s+4/eL/qm/iHoa8f8AgV/yGNV/694//QjXm1f40PmfWZcrZVi/+3fzPaqKKK6z5sKKKKACvDvhz/yWW/8A+3r/ANDFe414b8O2C/GS/LEAf6VyT/tiuSv8cPU+kyf/AHTF/wCD/M9g1HQNO1GOczWduZ5kK+c0QLA4wDn2qhcaWdItdKOnW/mfY5QsixR4LKRhmwPzre86P/nov/fQo86P/nov/fQr0Y1px0buux8pPDQldpWff8TJ1jSb/Up4jBqEdvFE6yIjW+87h77hTL60urvWdJimUyR2uZ5pgm1S4GAAPqSfwrZ86P8A56L/AN9Cjzo/+ei/99CiNaSSXa/4hLDxk29dbX+TKeu/8i7qP/XrL/6Aa8f+BX/Iwal/15p/6HXr2uSxnw7qOHX/AI9Ze/8AsGvIfgV/yMGpf9eaf+h15tX+PA+vy7/kU4v/ALd/M9D8beELjxS1mba6ig+zhwfMUnO7Hp9K5X/hUeof9BO2/wC+Gr1aitpUITd2fD1suw9abqTWr8z5/j8Nahc+IrnR7CMXM9vIyOw4UY7knp+Nep+FPAdn4fUz3ey9vCQyyNHxF7L+Peumt7G2tHle2gSNpnLyMo5Zj1JNT1FPDxg7vcywuWUqEud6vp5GV4o/5FHV/wDrym/9ANeVfAj/AJC+r/8AXvF/6E1eq+KP+RR1f/rym/8AQDXi3wk8SaV4b1HUZdauxbJPBGsZKk7iCc9AfWs6zSrwbP0DLKc6mUYqEFdtx0Xqj34jIIPQ1l3+jLNpCWenlbUwurxEZwpDZ/Xn86xv+Fo+D/8AoMp/36f/AOJo/wCFo+D/APoMp/36f/4muuOIjF3Ul958/PKsXNNSoy7fC/8AI6C80q01JEF/F5u3HAkYDI5zgH1qG60ppriwEUgW1tZDI6MSxc4OBk+h5rF/4Wj4P/6DKf8Afp//AImj/haPg/8A6DKf9+n/APiaaxMV9r8RPKcU96Mtf7r/AMib4kf8k61j/riP/QlrnPgh/wAinff9fp/9AWk8b/EDwzq3gvUrHT9TWa5mjCxoI3G47ge49qX4If8AIp33/X6f/QFrj5oyxCcXfQ+gVCrRySpGrFxfOt1bseiXVnbX0Hk3tvHcRZB2SoGGfXBql/wjWh/9Aix/8B1/wrTorscU90fJSpwk7ySZw2t/DuDVvEUE1usFjp8cSiVIUCmQ7mJxjpxjmuv0/TLPSrUW2nW6QRA52oOp9T61aoqI04xbaRlTw1KlJzitWFeHfFb/AJKhp/8A1yg/9GGvca8N+K3/ACVDT/8ArlB/6MNYYv8Ah/M+s4b/AN9f+Fntl1ZW19EIr23juIwchZUDDPrg1mW3hmwtNTubuCCFPOiEaokQGzrnB981redH/wA9F/76FHnR/wDPRf8AvoV3RqTimk9GfMSoxlJSlHVGNo8GoL4dWy3fZbm3YwrJJFuVlU8EDIzke9NtLDUNE0i8BuBfHa7xRxwbGLnJ9Tnmtvzo/wDnov8A30KPOj/56L/30Kt1m29rN3M44ZRUbXulZf1sUdAsG03QrW1k/wBYq5f6k5P868d17/k4GH/r+t//AEBa9w86P/nov/fQrw7XSG/aAgKkEfbrfkf7i1wYyTnaT6s+p4bpqnKrFbKnL9D3Siiiuk+dCiiigDi/iz/yTi+/34v/AEMVU+DqLJ8PSjjKtdSgj1HFW/iz/wAk4vv9+L/0MVV+DX/Igj/r6k/pXJ/zE/I+kj/yIX/18/Q6OXwza/a7F7YNFFasSU81+RjAA54qCDTN97rem3KMIL0CVHGcEMu0jPqCv610VFel7edtXf8A4e58m8PT+yrf8Nb8jBXQ7yeLT7fULmJ4LFlfMYO6Yr90nPT9aXQIZH1LVr+WNoxcXASMMMZVBtz+PNbtFDrScWn1/wA7iWHipRa6f5WX5nlvx0/5Aek/9fbf+gGum+GH/JNtI/65v/6G1cz8dP8AkB6T/wBfbf8AoBrpvhh/yTbSP+ub/wDobV5kf95l6f5H2OI/5EFH/G//AG46yiiiuw+XCiiigDnfH/8AyT3W/wDr0f8AlXI/Az/kXtU/6/B/6AK67x//AMk91v8A69H/AJVyPwM/5F7VP+vwf+gCuSX+8R9D6TD/APIkrf4l+h3WoaCl9qH21b27t5hF5Y8mQAY69CD1rGXTp9B8NWUl3JLI9rcq7qXDBAWxwQOnOa6+kdFkUq6hlPUEZBr04YiSSi9UfI1MNCTclo7P8bf5GJ4kS/vtPFtptkLlZNj+b5yqBhgcYPXgVW195rvS9PspoPIubq6jHlBg5UK24nI9hXSgAAADAHQCmlEZ1ZlUsv3SRyKIVuW2mzuFTD8/M7/EreVv6b6nA/Gj/kQR/wBfkX9af8Gv+SfJ/wBfUv8AOmfGj/kQR/1+Rf1p/wAGv+SfJ/19S/zry/8AmK+R9k/+RB/3E/QPi1/yLtn/ANfQ/wDQWryWvorU9HsNZt1g1S2S5jRt6q+eD0zxWZ/wgfhn/oDwfm3+NKth5VJ8yZ+cY7LKuJrOpFqx4RRXqHjnwbY22l2o8P6UFupblYz5QJOCD19BnHNWfDHwztrHFzr2y6n6rCpPlp6H3P6VzfV583KeV/ZeIdX2S+/oY3w803xMLkTW0r2undGFwpKtyD8q5HPvWj8bP+RLtf8Ar+T/ANAevRQAqgKAABgAdq86+Nn/ACJdr/1/J/6A9dc6fs6DVz9A4bwyw2MowTb1Nf4Wf8k30z6P/wChtUuu6dqQs9XuGMNwJ49sSjdujQdgMdfWovhZ/wAk30z6P/6G1dfXXhajpxi12RGcUlVxdaL/AJpfmzn0LxeItLN3tVpLN41AJxuBU4Gfb+VQ69PH/wAJPozfMVt5JDKQhITIGM8VvXNjBdyQSTqS9u/mRsDgqcY/rViulVkmpW6Nfff/ADPKlh5SjKF9G0/ut/kYmofv/FGkiI5MaSyPj+6QAP1rzf47f8fWjf7k381r1tbGBdRe+CkzvGIyxY8KDnAHavJPjt/x9aN/uTfzWuHGyToWXT/O59NwzBxzWMn1v/6Tb9D1XQ/+Re07/r1i/wDQBWX4+/5EXUv9xf8A0Na1ND/5F7Tv+vWL/wBAFSalp1vquny2V6heCYAOoYjODnqPpTa5oWXY8TGQdRVILd3R85Yor2z/AIVr4a/59Jf+/wC/+NZPinwJoWmeF7+9sraRZ4YtyEyscHI7Zrz3hZpXPj55RiIRcm1p/XY8prqvBOq+IYNUig0ZZbuAMBJbsT5ag9yf4frVvwt8O73VjFd6oDa2eQfLYYklXrx6D6816tpmlWWj2a22nW6wxqO3U/U9TTo0Jt82xrl+X15SVVvlX4s5v4pZ/wCFaanuxnEecf8AXRawvgb/AMivqX/X7/7TWt74qf8AJNdU+kf/AKMWsH4G/wDIr6l/1+/+01reX+8r0P1Oj/yIKn+Nf+2nU+IYb+a9Eg04XNpbwsyHz1X5yPvEH0HSsiwldtC8Mm5i8qNb0gEuCG4fB/Ou7IBBBGQeoNUdR0m31KxFq48pUcPGyDGxgcgivZp4hKKhJfP7/wDM+JrYVyk5xetnpp5f5GP43u7aLR1ikniSYzwtsZwGxvHOPSrHiCVLnR7ZrORZWe5i8to2zn5hnBH41ry2VtOwa4t4pXAxueME1Xn0mKa8s5QxjjtGLpAigKWIxk/TNRCrBKK7Nv8Ar7iqlCcnN/zJL89fxON+NP8AyIaf9fsf8mrQ+FP/ACTfTv8Ael/9GNWf8af+RDT/AK/Y/wCTVofCn/km+nf70v8A6MavKX+8v0PsZ/8AIhj/ANfP0Z2NFFFdZ82FFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHy7rV7c6f401W4sbiW2mW8mAkicqwBc55FM/4S3xD/0HdQ/8CW/xrr/B9rBefGy9hu4I54jNdkpKgZScnsa9l/4R7Rf+gRYf+Ayf4V5NOhKom1K2p+k47OMPgpQpVKXM+VO+n+R81/8ACW+If+g7qH/gS3+NH/CW+If+g7qH/gS3+NfSn/CPaL/0CLD/AMBk/wAKP+Ee0X/oEWH/AIDJ/hWv1Sp/McH+suD/AOgf8v8AI+a/+Et8Q/8AQd1D/wACW/xpG8V6+6lX1u/ZWGCDcNyPzr6V/wCEe0X/AKBFh/4DJ/hVfUNA0ddNuWXSbEEQuQRbJxwfal9Un/MOPEmEbS+r/l/keZ/AoYv9YA/55RfzavZa8Z+A/wDx+at/1xh/m1ezV0YT+CjxOI/+RnU+X5IKKKK6j58K+WfFOf8AhMNXxnP22bp/vmvqavC/CIDfHe5DAEfabvg/8Crixceblj3Z9bw3W+rrEVrX5Y3+485+f/b/AFo+f/b/AFr61dYIoy8ojRF6swAApImtZo/MhMMif3kwR+lZ/UHa/N+B3f63q9vY/wDk3/APkv5/9v8AWj5/9v8AWvq9LmwktFulkhEDfdkbCg/nRLc2EMCTSPCInYIrjBBJ4AyKf9nyva/4E/64wtf2X/k3/APlD5/9v9a9R+Bn/Ia1b/r3T/0I16/fRR/2fcfu1/1Tfwj0NeP/AAK/5DGq/wDXvH/6EaiND2NaOt7m9bNv7TyzEPk5eXl633foux7VRRRXpn5+FFFFABXynrmf+Ej1LGc/a5en++a+rK8N+HahvjJfhgCP9K4I/wBsVw4uPO4x7n1vDlf6tDEVrX5Yp2+883+f/b/Wj5/9v9a+t/Jj/wCea/8AfIo8mP8A55r/AN8is/qP978Dv/1uX/Pn/wAm/wDtT5I+f/b/AFo+f/b/AFr62VIGJCrGSpwQAODQyQopZ1RVAySQABR9Rf8AN+Af63r/AJ8/+Tf8A+Sfn/2/1r034G/8jJqn/Xov/odet62kTeG9QZFQg2kpBAHPyGvIvgV/yMGpf9eaf+h1MaPsa0Ve5vVzZZnleIkocvLbrfd+iPb6KKK9Q/PAooooAyvFH/Io6v8A9eU3/oBr5t8PeGdV8TSSxaLbCd4EVpAZFXAPA6kelfSXij/kUdX/AOvKb/0A15V8CP8AkL6v/wBe8X/oTVwYiCnVjFn2WSYqeEy3EV6e8Wt/uOf/AOFU+MP+gWv/AIEx/wDxVH/CqfGH/QLX/wACY/8A4qvoC+1Oz01Fa9nEQc4UYJJ/Ac02TVrKPTft/nBrbjDoCc5OAMfU1qsvi0mr6/12OZ8X4tNpqGnr/wDJHgP/AAqnxh/0C1/8CY//AIqj/hVPjD/oFr/4Ex//ABVe+Xus2GneX9tuBEZBuVSpJx9AOKe2p2qTWsRkybvPksBlWwM9fpR/Z8bX1/r5C/1vxd7Wh+P+Z866n8O/E2j6bNf6hYLFbQLukcTo2BnHQHPevS/gh/yKd9/1+n/0Ba6P4kf8k61j/riP/QlrnPgh/wAinff9fp/9AWsYUo0q6Uex24rMauYZPUqVUk1JLT5ebPSqKKK9A+KCiiigArwT40f8j3H/ANeUf/oTV73XhvxW/wCSoaf/ANcoP/Rhrkxn8P5n0vDLtj7/AN1nm/z/AO3+tHz/AO3+tfW/kx/881/75FRNJZrOIWeASnpGSNx/CsfqDe0vwPWfF8VvR/8AJv8AgHyb8/8At/rR8/8At/rX1i81nHdx2zmMTSglE28kDqaUS2RnMIeAyjrHld35Uf2fLv8AgL/XCP8Az5/8m/4B8m/P/t/rW14Nz/wnWi5zn7bH1+tfTEMtpcSSxw+W7wtskUL904zivE9dAX9oCAKAB9ut+B/uLWdTCui02+qO/B58syjVpKny2jJ3vfbTsj3SiiivVPzYKKKKAOL+LP8AyTi+/wB+L/0MV4PZ69q2nW/kWGp3drFkt5cUxVcnvgV7x8Wf+ScX3+/F/wChisr4R6Tp174GEt5YWtxJ9qkG+WFWOOOMkV51am6lflTtofb5TjKeCymVWpDmXPa3yR5H/wAJb4h/6Duof+BLf40f8Jb4h/6Duof+BLf419Kf8I9ov/QIsP8AwGT/AAo/4R7Rf+gRYf8AgMn+FH1Sp/MX/rLg/wDoH/L/ACPmv/hLfEP/AEHdQ/8AAlv8aP8AhLfEP/Qd1D/wJb/GvpT/AIR7Rf8AoEWH/gMn+FH/AAj2i/8AQIsP/AZP8KPqlT+YP9ZcH/0D/l/kfMN9rOpaoiJqOoXN2qHcqzSlwp9RmvoP4Yf8k20j/rm//obVx/xq02xsdF0trGyt7ZmumDGGJUJGw8HArsPhh/yTbSP+ub/+htRh4OFdxbvp/kTnOKhi8pp1qceVOe3ykdZRRRXonw4UUUUAc74//wCSe63/ANej/wAq8L8L+PdW8I2c9tpcdq6TyeY/nozHOAOMEele6eP/APknut/9ej/yryj4b+ANL8YaXe3OpT3UbwTiNRA4AI2g85B9a8/EKbrJQ3sfbZJUw1PLassUrw5l59g/4XV4n/54ab/35f8A+Lo/4XV4n/54ab/35f8A+Lrsv+FJeHP+fvUf+/q//E0f8KS8Of8AP3qP/f1f/ian2eK7m/17h7/n3/5Kcb/wurxP/wA8NN/78v8A/F0f8Lq8T/8APDTf+/L/APxddl/wpLw5/wA/eo/9/V/+Jo/4Ul4c/wCfvUf+/q//ABNHs8V3D69w9/z7/wDJTzbxL8Rta8VaSNO1KKzSHzFkzDGytkdOrH1r1P4Nf8k+T/r6l/nXG/EH4b6R4U8MjUdOuLySb7QkWJnUrg5z0A9K7L4Nf8k+T/r6l/nRRU1XtPews1q4SrlHNhFaHOultbM72iiivSPgwooooAK85+Nn/Il2v/X8n/oD16NXnPxs/wCRLtf+v5P/AEB6wxH8KR62S/8AIwpepg+DPilo3h3wnZ6XeWt7JNAG3NEilTlieMsPWt3/AIXb4e/58tR/79p/8VXCeHfhVqniTQrfVbXULOGKfO1JA24YJHOB7Vp/8KO1v/oK2H/fL/4VxQnieVcq0Pq8ThshlXm6tT3ru+r3vr0Oo/4Xb4e/58tR/wC/af8AxVH/AAu3w9/z5aj/AN+0/wDiq5f/AIUdrf8A0FbD/vl/8KP+FHa3/wBBWw/75f8Awq+fFdjD6rw7/wA/Pxf+R1H/AAu3w9/z5aj/AN+0/wDiq4H4k+NLDxjNYPpsNxELZXD+eoGd2MYwT6Vrf8KO1v8A6Cth/wB8v/hXLeMPBV74NktUvrqC4N0rFTCDxtx1z9ayqyruDU1oehluHyaGKjLCzvPW2r7a9Ox9FaH/AMi9p3/XrF/6AKvVR0P/AJF7Tv8Ar1i/9AFXq9WOyPzqt/El6sKCAwwRkehoopmQUUUUAch8VP8AkmuqfSP/ANGLXA/C7xvofhfQ7221m5khlluvMQLCz5XYo6geoNd98VP+Sa6p9I//AEYteH6F4N13xLay3Oi2azxRSeW7GVUw2Accn0IrzsRKca6cFd2PucmoYevlM6eJlyx597pdI9We0/8AC3fB/wDz/wA3/gLJ/hR/wt3wf/z/AM3/AICyf4V5V/wqnxj/ANAtP/AmP/Gj/hVPjH/oFp/4Ex/41Pt8T/L+DNP7HyP/AJ//APk8f8j1X/hbvg//AJ/5v/AWT/Cj/hbvg/8A5/5v/AWT/CvKv+FU+Mf+gWn/AIEx/wCNH/CqfGP/AEC0/wDAmP8Axo9vif5fwYf2Pkf/AD//APJ4/wCR03xL8e6B4l8KrY6RdSSzi5SQq0LoNoBzyR712nwp/wCSb6d/vS/+jGrxTW/A/iDw7p4vdYslgty4jDCZW+Y9BgH2r2v4U/8AJN9O/wB6X/0Y1VQlOVduas7GWcYfDYfKowwsuaPPvdPWz6o7GiiivRPhwooooAKKKKACiiigAooooAKKKKACiiigAooooA8N8D/8lyvP+ut3/M169rUMciI1xeXlvEoOFtdwLH3KjNeQ+B/+S5Xn/XW7/ma9kvzqiuh0xbWRcYdZ2ZcH1BAP5YrDAaJ+rPoOJ9a0P8Ef1Oai1K8m8Dwg3LSTXNwLZJt3zhS+ATj+LFadzDNqWu/2eLye2trW3VyIHKM7EkDLdcACqeo6Pc2HhySZcT3S3i3sixjC5BBIX2wK0rmzv/7SXVNIaBzNAI5IrgsoI6hgQDzzXsylD4o21b+/T+kfFKNRWjK7so/dd/8AAujH/tCeTR2uJZpDLpN/5bPu/wBagbbz68Guovzu0m5I7wOf/HTXP3umiy0mPTTmS41O7DzFRwCW3MfoMV0F+Nuk3KjtA4/8dNc+JcXG8fP9P1udWCU41LS8vzf6WPIvgP8A8fmrf9cYf5tXs9eMfAf/AI/NW/64w/zavZ68vCfwUfV8Sf8AIzqfL8kFFFFdR8+FeGeEP+S8XP8A183f/s1e514Z4Q/5Lxc/9fN3/wCzVyYj4oep9Lkv+74v/A/1PWNeRbnUtKtJxugkmYuh6PheAR3FYesTHSL7WYNPUQQmwWQJGNqo5JXIA6cV1WqaXDqsEaSvJE8TiSKWI4ZG9RVV/D8Tabe28k8s814hWS4lwXPGAOABgV7FKtCMVzfd873Piq1CpKbcfVP5Wt/XcoNBFL4g0zS54kktI7FpFjcBlLAqAcfQmqt7bRx6f4isoVCW8KiaNVHCNt3cenIzWuukyX+nWLXcktnfW6gebCy7hxgjoQQeuKiv9Ekj8Pz2NgXmmu3AmmlYbiGPzMenbsKuNSKklf8Aq97mbpScG+Xz87ctrW9S7HM1x4ZWZ/vSWm4/ileTfAr/AJDGq/8AXvH/AOhGvYLiJYNHliT7qQFR+C14/wDAr/kMar/17x/+hGvGrtPEQa8z7XK01lGKT3tD8z2qiiiuk+eCiiigArw74c/8llv/APt6/wDQxXuNeHfDn/kst/8A9vX/AKGK5K/xw9T6TJ/90xf+D/M9b8TXctpocjQMyPI6RB16ruYLkfnVW4gTR9Y01rVpAk26GSMuSG+XIY578VralYRanp8tpPkLIOGHVT2I+hqrDpMxvYbnULz7S0CFI1EWwZPViMnJr1ac4qFm+/z00PkKtOcql0u1vKz1MC2Mz+E4JPNkjl1S8BlkQ4YAtyM/RcVfmiSx8UWtjDuNtfW0iyRMxIyuOee/NJp2hzzeHX0q9LW7W1wTBMg7A7lYfyrRTTZIbptRu5vtdzHCY4gkW0AdTgZPJredSCk9e/47fcctOlUaTt/Lr6P3vvMuzlz4F1K35xaRXFuCT1ChgP0xXnHwK/5GDUv+vNP/AEOvTY7SS18C332iMxzzW880qH+FmDHFeZfAr/kYNS/680/9Drx8Y08XFrzPtslTWR4lS/u/me30UUVufPBRRRQBleKP+RR1f/rym/8AQDXlXwI/5C+r/wDXvF/6E1eq+KP+RR1f/rym/wDQDXlXwI/5C+r/APXvF/6E1cdX/eIfM+oy/wD5E2K9Y/mj0pDv+IUol52WKmIHtljux+lZFiFa4uLZQPsjaxhQPu8IDgf8Crq7/SbTUmRrlG3x/dkjcow9tykHHtVbUNGWXR1tNN227xOskR7Ag55+vNe1CtDRei9LdT4aph5+899W/W6tYraf+88Waz9oAJRIlTd2Qrk49s1iWTSJ4Tt5gCWh1Ei3Popkxx7YJrqbzRrTUXjlvI2EyptLRSsmR3GQRkfWmXemSTXWnpB5cdlavvePHLEDCgDpjPNEa0NPl+C/UVTDzadv71v+3v8AL9DL+JH/ACTrWP8AriP/AEJa5z4If8inff8AX6f/AEBa6P4kf8k61j/riP8A0Ja5z4If8inff9fp/wDQFrxpf7yvQ+2o/wDIjq/41+SPSqKKK6z5sKKKKACvDvit/wAlQ0//AK5Qf+jDXuNeHfFb/kqGn/8AXKD/ANGGuTF/w/mfScN/76/8LPcHOEYjqBXDQQRXXw/utUnVWviJJvtGP3isDwA3UY9K7qsU+GLUyOonnW0kk817MEeWzdfTOM84zivVoVIwTvpqn93Q+QxFKU2mtdGreuz+X6lO0mefUry9cZmt7CMJz0LAsf1A/Kqa2sH/AAhdpqqwx/biI5zOFG9nLDOT1PWtw2dxD4lFxEga0nt/LmGR8rKflOO/BIpkXhqCOZc3Vy9tHL5qWrMPLRs5GOM4HpmtfaxVnfs/u3Rz+xm01buvm7WZXtGMXjq4ReBc2KSuP9oNjP5fyry3Xv8Ak4GH/r+t/wD0Ba9dsrCf/hI7/UblQqsqwwDOcoOSfzJryLXv+TgYf+v63/8AQFry8wafJbyPr+FotVMRf+Sf5r/hz3OiiitTwwooooA4v4s/8k4vv9+L/wBDFVfg5n/hX/HJ+1S4/SrXxZ/5Jxff78X/AKGKq/Bw4+H+euLqXp+Fcn/MT8j6SP8AyIX/ANfP0OmbUdRtb6yjvYbfy7pyh8pjujOCR169PamnWSl9qcrtiz0+MBhj7z43Hn2GPzqpp7S32tDUdUs7uGSMmO2hMTFY1PViemT/ACqBLFrtPE2lrhZp5N6ZHUMgwfzBFe17OF7S7a/f+iPivaTavF7vT7n+bRd/t29gj064vbaJYL51QhCd0Jb7ufX9KtaLqE91Nf2t4VM1pOUyBjcp5U4+hrLlhvdUstIszZywNbyxyXLSLhU2Y4B75x2qzoP77xDrtynMbTJED6lVwf1onCHJLTX/AIKt+FxU6lTnhduzt/6S2/yXocb8dP8AkB6T/wBfbf8AoBrpvhh/yTbSP+ub/wDobVzPx0/5Aek/9fbf+gGum+GH/JNtI/65v/6G1eJH/eZen+R91iP+RBR/xv8A9uOsooorsPlwooooA53x/wD8k91v/r0f+Vcj8DP+Re1T/r8H/oArrvH/APyT3W/+vR/5VyPwM/5F7VP+vwf+gCuSX+8R9D6TD/8AIkrf4l+h6LdvfiUCzSDywhJeUk8+mB/Oss67Pd+GYby0jEV1cSCJU+8A27BPv0JqTxBPeN5djZxXCxz/AOuuYoy3lr6DH8RqDUPs9pp2nSxRPBbWlyilZIypAPGcH3PWvXpwXLG61b/q/qfG1py5pWdkk/6Xp/XlcfULuTUpLGxELvbRK00kueWPQAD6Zqk/iC4Nnp97tWJGuvs13CedpJ28H2NShJdM8RX1z9mmnhvI0ZWhQttZRjBx61lX1pNB4bgt7hdl1f6gsnlZztJk3Y/AVpThBtK29vxWv3GVWpUjGTT25vw2+/8AG7KXxo/5EEf9fkX9af8ABr/knyf9fUv86Z8aP+RBH/X5F/Wn/Br/AJJ8n/X1L/OvD/5ivkfdv/kQf9xP0O9ooorsPlwooooAK85+Nn/Il2v/AF/J/wCgPXo1ec/Gz/kS7X/r+T/0B6wxH8KR62S/8jCl6mv8LP8Akm+mfR//AENq277XYrCZxNa3TRR48y4WP92g9Sc/yzWJ8LP+Sb6Z9H/9Daruu6hHfXB0OCWMeYMXkjOB5MZ7c9zXTg4qaimtLHDnM3DFVmnZ80vzehp3mqLbzWUUMfnNeSbVw2MLjJb/AD61Bc+Iba2uJEMFxJFCQJrhEBjjPuc9u+KgZUj8U6eqNmIWcixEHIJyn9Ky4po4PC+t2M7gXZknAiJ+Zi+dmB3zkV1wpQaWn9N2/A8edaabV/6Ub/j+SOii1PzNZaxMWFMImilDZEgzg/lXlfx2/wCPrRv9yb+a13cUE1vfeHYXBE0cLiUZ6DaOv41wnx2/4+tG/wByb+a15+YQjGl7v9a2PpuFakqmZQ5l3/GF/wALnquh/wDIvad/16xf+gCr1UdD/wCRe07/AK9Yv/QBV6qjsjzq38SXqwooopmQUUUUAch8VP8AkmuqfSP/ANGLWD8Df+RX1L/r9/8Aaa1vfFT/AJJrqn0j/wDRi1g/A3/kV9S/6/f/AGmtccv95XofT0f+RBV/xr/209Cu9TtLFttzLtIXcQEZsD1OAcCmz6vZwaat/wCaJYHxsaP5t+TgYqHW9RWythDGokvLkGOCLGdzH19h3rGudN/s2z8O6fM+9VvMuccFtrt+WTXrU6UZJN/8Oup8ZVrTg2o62X3PS1zoLnVrK0mEM822Upv8tVLMF9SADgU1tXtVuLONWLreZ8mVRlCQM4z+FUdPZB4p1gSkeZtiIz/c2/yzmsK082PwjDKoO5dSJt/90y9vbGauNCL/AA/FfoZTxU4pv/F/5K/1K/xp/wCRDT/r9j/k1aHwp/5Jvp3+9L/6Mas/40/8iGn/AF+x/wAmrQ+FP/JN9O/3pf8A0Y1eOv8AeX6H2s/+RDH/AK+fozsaKKK6z5sKKKKACiiigAooooAKKKKACiiigAooooAKKKKAPDfA/wDyXK8/663f8zXuVfOt3beKdG8c6jqWi6bqEc32qby5ls2cFWY9MqQcir//AAl3xO/556l/4LP/ALCvNo1lSTi09z7vM8qnj6kKtOpFLlS1f/AZ73RXgn/CXfE7/nnqX/gs/wDsKP8AhLvid/zz1L/wWf8A2FbfW49meX/q3iP+fsPvf+R73VbUf+QXdf8AXF//AEE14b/wl3xO/wCeepf+Cz/7CmSeKviZLE8ckWpFHUqw/s3qD/wCh4uNtmVHhyupJ+1h97/yNf4D/wDH5q3/AFxh/m1ez15J8FdKv9OvNVN/YXNorRRBDPCybsFumRzXrdXhVakjl4ilGWZVHF3Wn5IKKKK6TwArwzwh/wAl4uf+vm7/APZq9zrwjUvBXjW38Y6hqmi2U0TPdSvFPFMinazH37g1yYm94tK9mfS5E6bhiKU5qPNGyu7bnu9FeG/2Z8WP+euo/wDgUn+NH9mfFj/nrqP/AIFJ/jR9Zf8AIxf2HD/oJp/+BHuVFeG/2Z8WP+euo/8AgUn+NH9mfFj/AJ66j/4FJ/jR9Zf8jD+w4f8AQTT/APAj2u+/5B1z/wBcm/ka8c+BX/IY1X/r3j/9CNVm0r4ruhV5NQKsMEG6TkfnXRfCbwnrXh3U9Rl1mxa1SaFFjJdTkhiexNZ88qlWL5WrHasPSwOXYin7aMnLlsk+zPUaKKK7z44KKKKACvDvhz/yWW//AO3r/wBDFe414JN4K8cWPia91DRrKeB5J5Sk0UyKSjMT69+K5MRdSjJK9mfTZG6cqWIpTmo80bK7t3Pe6K8N/sz4sf8APXUf/ApP8aP7M+LH/PXUf/ApP8aPrL/kZP8AYcP+gmn/AOBHuVFeG/2Z8WP+euo/+BSf40f2Z8WP+euo/wDgUn+NH1l/yMP7Dh/0E0//AAI9j13/AJF3Uf8Ar1l/9ANeP/Ar/kYNS/680/8AQ6il0f4qzQvFK+oPG6lWU3SYIPBHWug+E3hHW/DutX8+sWJtYpbdUQl1OSGzjgmsuaVStF8rVj0I0KOByzEU3WjJytaz7M9Uooor0D4sKKKKAMrxR/yKOr/9eU3/AKAa8g+C2p2Omapqj6jeQWqyW8QQzSBAxDN0zXtOq2R1LRryxVxGbmB4g5Gdu5SM4/GvIv8AhRN7gD+3bc4/6dW/+Krjrxqe0jOCvY+oymvg/qVbDYmpyc7XRvb5HqX/AAlXh/8A6Den/wDgUn+NH/CVeH/+g3p//gUn+NeW/wDCiL3/AKDtt/4Cn/4qj/hRF7/0Hbb/AMBT/wDFUe1xH8n4h9Qyb/oKf/gL/wAj1L/hKvD/AP0G9P8A/ApP8aP+Eq8P/wDQb0//AMCk/wAa8t/4URe/9B22/wDAU/8AxVH/AAoi9/6Dtt/4Cn/4qj2uI/k/EPqGTf8AQU//AAF/5HX/ABA8Q6NeeAtVgtNWsppniASOO4VmY7h0ANZ3wQ/5FO+/6/T/AOgLWD/wom9/6Dtv/wCAp/8Aiq9A8BeEZfB2jz2U94l20s5l3pGUA4AxjJ9KUFVlWU5xtoaYqeX0MtnhsNV525J7NfodRRRRXafJhRRRQAV4d8Vv+Soaf/1yg/8ARhr3GvIvib4P8Q614wiv9FsWniS3RRIsirhgxPc+4rlxSbp6I+h4eqU6eMvUkorlerdj12ivDf7M+LH/AD11H/wKT/Gj+zPix/z11H/wKT/Gl9Zf8jL/ALDh/wBBNP8A8CPcqK8N/sz4sf8APXUf/ApP8aP7M+LH/PXUf/ApP8aPrL/kYf2HD/oJp/8AgR7lXhmvf8nAw/8AX9b/APoC0v8AZnxY/wCeuo/+BSf403RPBXjJ/HGnatrVlM5S6jknnlmRjhccnn0FY1akqtkovc9PL8HRwDq1JYiErwasme6UUUV6J8QFFFFAHF/Fn/knF9/vxf8AoYqr8Gv+RBH/AF9Sf0rQ+KFpcXvw/vYLOCW4lZ4yI4kLMcOOwryPR9T8faBYfYtJs9St7feX2f2eW5PU5K1w1Z+zr8zXQ+vwGFeMyh0ISSfPfV26I+iqjFvCLo3AjUTFdhfuR1x+teEf8Jd8Tv8AnnqX/gs/+wo/4S74nf8APPUv/BZ/9hV/XIrozmfDVd71Yfe/8j3uore1gtEKW0SxqzFiFHUk5Jrwn/hLvid/zz1L/wAFn/2FH/CXfE7/AJ56l/4LP/sKPrkbWsw/1ar3v7WH3v8AyOp+On/ID0n/AK+2/wDQDXTfDD/km2kf9c3/APQ2rxvXLrxz4kgih1qx1K5jhfeg+wFcHGM8KO1e0/Di2ns/h9pcF3DJBMiNujkUqy/O3UGs6Mueu5JdP8jrzPD/AFTKKeHlJOSl0d/5jp6KKK7z44KKKKAOd8f/APJPdb/69H/lXI/Az/kXtU/6/B/6AK6/x8Cfh/rYAyfsj8D6V494D+IA8GaddWr6XJdm4mEu5Zdm35QMdD6VxVZxhXjKXY+sy7DVcVlFalRV5OS/TufQVR3FtDdwNDcxrLE33lYcHvXlf/C8o/8AoXpv/Akf/E0f8Lyj/wChem/8CR/8TWqxVJO6l+Z5z4fzJqzpfjH/ADPV1UKoVRgAYAqOW1gmmilliV3hJMbEfdJGMivLP+F5R/8AQvTf+BI/+Jo/4XlH/wBC9N/4Ej/4mj61SWvN+Yf6vZi1Z0vxj/mbHxo/5EEf9fkX9af8Gv8Aknyf9fUv864Lxt8Sl8XeHxpqaTJakTLL5hl39M8Y2j1rvvg2pX4fJuBH+lS9R71hCcZ4i8ex62KwtbCZH7KsrS579O3kd5RRRXefHBRRRQAV5z8bP+RLtf8Ar+T/ANAevRq85+Nn/Il2v/X8n/oD1hiP4Uj1sl/5GFL1Nf4Wf8k30z6P/wChtXRTaNplzM0txp1rLI3Jd4VJP4kV5z4C+IHhzRPBVjYalfmK5hD708pzjLkjkD0NdH/wtXwh/wBBQ/8Afh/8KKNeMIK0radzXMctxVXF1WqMmnJ/Zff0OgutIinlsZICLc2T7owijG3GCuPQ1PJp1lLdrdS2kD3C42ytGCwx0561zH/C1fCH/QUP/fh/8KP+Fq+EP+gof+/D/wCFa/WY/wA/4nB/ZGLu37CX/gL/AMjo007GtvqMkzOTEIkjxwgzkn8a8p+O3/H1o3+5N/Na7P8A4Wr4Q/6Ch/78P/hXmvxX8UaT4mn01tGuftAgSQSfIy4ztx1Hsa5sVWjOk1e+x7vD+X4mhmEJzpyS1u2n2Z7Zof8AyL2nf9esX/oAq9VHQ/8AkXtO/wCvWL/0AVerqjsj5ut/El6sKKKKZkFFFFAHIfFT/kmuqfSP/wBGLWD8Df8AkV9S/wCv3/2mtb3xU/5Jrqn0j/8ARi1yfwb1rS9M8OX8epaja2kj3m5VnmVCRsUZAJrik0sSr9j6nDQlPIqiirvn/wDkT1C50yzu7hZ7iENKg2q+4ggenBqtqeiJe6fFBbuYHglWaJ8ltrA/XPIyPxpn/CW+Hf8AoO6b/wCBSf40f8Jb4d/6Dum/+BSf413RruLTUtj5mWBqSTTpvXfRly40uzu5hNcW6tLt2bxkEj0yO1Q3elm4uLDy3WK1tH3mEL97Awo9gOtQ/wDCW+Hf+g7pv/gUn+NH/CW+Hf8AoO6b/wCBSf40Ktb7Q3gqj/5dvXyZyvxp/wCRDT/r9j/k1aHwp/5Jvp3+9L/6Mauf+Lmu6TqXgtINP1O0upftcbeXDOrtjDc4BroPhT/yTfTv96X/ANGNXFFp4l27H0daEoZFFSVn7T9GdjRRRXYfMBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRXO3nii4h1q8sLPSJLsWUSyzSLOq4Vs9AeT0NRKcY7mtKjOq2odPNL8zoqK5y48Sy3tnpcnhwQyNqMhCvcqwVFUHdkDBzkYroU3+Wvmbd+Pm29M1Saew6lGdJLn0vfTrpoOorK1LWzaXqWFjatfX7p5n2dZAm1AcFix4HJHFXbKa4ntVkvLX7LKesXmB8fiOKSknsKVKcYqb2fpf7t/mWKKKKoyCiiigAormdf1DWdC0uTUnuLaUJcIotxEQCjOFxnOd3NaV74j0zTn8u9uPLl2B/KCMz4PfABJ6VHPHrodP1Wo0pQ9699vK3+aNSis9td01dFXVjdKbJlDCUAnOTjGMZzntjNZen6/FqvitY7C6aS1FqWaIqVKvuxypAIp8yul3/AMrijhqsoylZpLfTt09TpKKp6lq1lpFuJtQm8pGbaoClmY+gABJrFu/E1p9r0y7hvSthKZRJlSu4qp42kZzntjNTKpGO7Cnh6tRXjF216dle3qdNRWdZa9puoWMt5bXS+RCxSVpFKbCOxDAEdabp3iLTNVuJILK4LSxruKPEyEr6jcBke4quaO1yXQqq94vTfTb1NOisU+L9DFxHD9uUtJJ5YIRiobdtwWxhTn1xVjU/EGm6RIkd9cFHcZCJG0jY9SFBwPc0ueNr3H9XrXUeR3fkzSorKvvEuk6c0S3N1zMm9fLRpPl/vHaDge5rP8VanPBa6RNpl0VW4v442aIgiRCGyPpxTckVTw1Scopq19m9tDpaKKKo5goorkp/EGuvqmsJp8Fi9tpbqGSTd5kg2Bjg5wDzUSmobm9GhKs2o207+tv1Otork5vGEN1qelW1pe29pFe27XEkkxXcgwNq4J4Jz+ldYOg5z704yUr2CrQqUUudWv8A8MFFYmtXOt2Sz3VrNpkdlCm8m4SQsABznBxVnw/e3uo6BaXmpwxw3E6bykRJUA9OvtihSTdglQkqXtbq2xpUUUVRgFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUjMqKWchQOpJxiuP8Spps3jCxi1uZI7T7HK37ycxjcGTHQjJ61z2pXd6/gyxhiimvLKXVAkMksgRpIgTtU56g+voK5/ba2t/V7fqerRy72sYPmtfy067O+r08j1FWV1DIwYHoQc0tYT+JNO0pI7a+hls5tgZoIrd5FTPbcikGsfxZqkdxa6Ve7ppNEaZhdxKrJJJx8oCnDH5uw5q5VIxVznpYOpUmotNJ7O2/XTo2+mup2tFcp4Cle4sb6eGZm057k/YonbLxIOobPI5zwea6uri+aKZjiKPsarp3vYKKKKowCiiigAoormtd1XxBpMV1qEdrZPp9swJjZm814xjcwOcA9eKmUlFXZtRoyrS5ItX8zpSM9absT+6v5VXfU7KKFZJ7qGFWQP+9kC4Hrz9alS5gktftKTxtBt3eaHBXHrnpindEcskr2H7E/ur+VGxP7q/lWGdfE/iuwsbCe3uLO4t5Xd42D/MpXAyD71tTzw20LTXMqQxL955GCqPxNJSTVy50qkOVPrr+Nv0HbE/ur+VGxP7q/lWFq3iEW91pBsbi3ltby6MUsoYMoUIxOCDgdK2ba8tryMyWdxFOgOC0ThgD6ZFCkm2uw50akIqTWj/AM7EmxP7q/lTgAOgx9KrW+pWN1M0NreW80qfeSOVWZfqAaRtV09JVje+tlkZtqqZlBJ6YAz1p3RHJO9rMtUVBdX1pYoHvbmG3VjgGWQICfxps+pWNrDHLc3tvDHKMxvJKqh/oSeaLoShJ2styzRWR4i1SbTdHS7smQs00SAkZBVnVT+hrWByoPtQndsbpyUFN7O/4W/zFrH8TeGLHxXpiWOptMsSSiUGFgpyAR3B9TWxWN4o1e50bSUuLFInmkuI4VE2do3sFycc96mfLy+9saYb2vto+xdpX0OW/wCFLeGf+e2of9/l/wDiaP8AhS3hn/ntqH/f5f8A4mtxPEt1p9xe2evi1juILY3UUsJKxunTB3HIOf51c8K6q+s6LDfS3cM8kyK7RQ4/ckjO04P86wjToydlE9eeOzSnB1HVdtNb73OX/wCFLeGf+e2of9/l/wDiaP8AhS3hn/ntqH/f5f8A4muy1zVk0XSnu3QyEMqIg/idjhR+ZFUNN1fUo9bXTNfS0SWeHzbZrXdhsffBBPUZFP2VG9uUmOYZnOm6iqu3r23t6dTnP+FLeGf+e2of9/l/+Jo/4Ur4Z/57ah/3+X/4mvQqKv2FL+U5/wC2Mw/5/P7yK1t0tLSG3izshRY1yecAYFS0UVseW227sKKKKBBRRRQBkeKdCHiXw3daSZzbi4C/vAu7bhgen4V5z/wolf8AoPt/4Cj/AOKr12isZ0adR3kj0sJmmMwcHToTsm77J/mjyL/hRK/9B9v/AAFH/wAVR/wolf8AoPt/4Cj/AOKr12io+q0ex2f6w5n/AM/Pwj/keRf8KJX/AKD7f+Ao/wDiqP8AhRK/9B9v/AUf/FV67RR9Vo9g/wBYcz/5+fhH/I8i/wCFEr/0H2/8BR/8VXovhXQB4Z8N2+lC4Nx5JY+YV25yxPT8a2KKuFGnTd4o5MXmuMxkFTrzur32S/JBRRRWx5gUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVxS217d+OvEEFjcxW/mWkCO7xlmAO8ZXkAH65rtaiS1gjuZLiOGNZpQA8gUBmA6Anv1rKdPna8v8AJo6cPX9ip6atW/FP9DkZ/Dmnadqnh+xSBZUTzFLSDLPxnJ/E5pb9/EC6hOLV9aEIc7BFBalMexY5x9a657eGWaOWSJGkjzsYrkrn0qSj2em9vQ3+vS0clzO3XXq2eemxa5+Isy6rdXdqkumRujiTyWZlwGyyn1PIHFb3gS7vLzwyHvppLgpNIkU0gw0kYPyt7/Wtq+0ux1NFTUbOG6VDlRLGGwfxqyiLGgRFCqowABgCiMHGV/X87/gViMYq1FQtrp8rXWnr19BaKKK1PNCiiigDirq9/tjxMDqcN/Bpunyfu7f7FKwuZP75wp+Udh6jNbVvbeZ4uubtrc7Gs4xHK8eO7ZAJ6HpkVt0Vkqe135/hY7Z4pOPLFWVrb/Pt16nApp97bWdrObSZrez1ieaS3WMktEWbaQvcDIPFatgzXnjmS9h06e3tzabPtEkJTzTu9Dzx711NFEaSi1rt/lYueNc07x1d197uc7rySQ+ItI1B7eW4tYBKsnlRmQxswG1toyexGR60Sqmpa1pVzHYyCBJJSTLAVwduAxBGRz6810VFP2d+vW5ksTaKVtUmvk7/AOZyOorqFlc+ILjTtO+0u8cTRxMmVkPRjjoxA/lWfp8GoT+NNNupJb+7hS1lV5J7MQLEx2/IPlBP48eneu+opez1TvsbRxzjFrlV2rX+Vv6tbzOPGmSL8P7qFbNhcu8r7BH8xbzWIOOucYpLUS6T4u1C81G0uJIbu2iEMscLS4woDIdoJHPPPFdjRSVO1rPb/KxP11tSUlpK/wCLT/Q4O8sb2PxJf3LTX2n2t9bxiJbeyWfIC4MZ+VtvPPpzViXSRZeGtBtkS/lFtfLIAYA8n8X3gDhR834cV2lFNU0v687lvHzairbW/BWXS+3e4UUUVqeaFcJa6NPq3irxJF/aVxaWrXEYliiVf3o8tcjcRkZ6cGu7pAoBJAAJ68daiUFJq504fESoKXLu1b01T/Q5K60yzj+IOkQpZw+UlhMADGCBgoB+VQT6DrL3MjJbTFS5Ixr0y8Z9NvH0rtMDdnAz60tT7JdTdY6aSstlbVvu30a7nKeK431C30zw6pZXvnBuMMW2woMv83fPA9811MaLHGqIMKoAAHYUuAWBwMjoaWrUbNvuc1Stz0400rJX+99fusvkFFFFUYBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYt7on23xRa388UMttDbyRski7juYqQcEexpPEmjTarZ2MNl5afZryOYhuBtXPAx9a26KhQS/M6I4mpGUZL7Oxl3+gW+oXRnlur+JiANsF5JGv5A4rM1Lw5dRX+l3ulkXn9niQeRfTMd27+LeckEfTvXT0UOEWOniqsLa3Xb1VvyMDw5odxp1/qWoXYiil1B1c20BykeAe+BknPJwK36KKqMVFWRnVqyrT55b/5aBRRRTMgooooAK4bxFrthrGuNoFxqMNjZWzK97JJIFMxBz5Qz29T+FdzTDBExJaJCT1JUVE4uWh1YatCjPnkrvp5PvszlZre01D4hWTtFFc2/9mO0ZKhlPzpg+nSsa/hlj0OWCCKT+z7bWf38UYOFgySeB/DnBI9K9FCqMYUDAwOKNowRgYPXjrWfsv1/NP8AQ6YY9wcdLpW69r/jqcPay6VcfErT5dEWMxfYpg8sCYjY5XgEcE+taXi3Caho0t5j+y47hjdl/wDVj5DtL+27HXvXSrGiY2oq46YFKyhlwwBHoRT9n7tr9b/jch4xe0jNJ6K2r1663tvrocPq66PqF9oI0uOCW0bUj5nkp+7Y7G9ODUp2aT4q182dmGhGmRzG2jXCyMCw6DuQAK7IRooAVFAHIwOlLtGScDJ4JxS9lre/9WsV9d93ls2rW1f96/8AwDgNOvILvXNFnsm0/L5EgsIGUxgpnY75wfocHjNQx6bay+BvEtxJao1wLm5ZJCnzAjlSD1HPpXoixon3UVfoKXauCNoweox1qXQurX7/AI2/yNP7Rs04xtt17Nvt5nDieyi16zuPEjRi2fS4xA91/q9+Tv68bsbfeoLuSzi8UGeWTT4NOls41snvoC0JGTuCcgA/0xXfNGjgB0VgOgIziho0ZQGRSB0BHStHTZMcck78r2tvp8tPv+Zx01tHaeCLaOW/SWL7ZHIszRuqlfODBQOTjHArsl+6MelBVSuCoI9MUtXGPLc461Z1Vr3b++3kFcv8QI3l8OwpE7Rub63w6gEr+8HODxXUUhAb7wB+tKceaNicPV9jVjUtezOan8PJaaTql1fXMmpXc1uymadF+VQMhQAMAZ5+tZun6RcT+CtD/sy1w/2VDL5d69oSdo5OwHd+NdwRkYNAAAwBgegqPZK7Z0xx1RRs9Xe/bpa1lY49NExo1/F4jLWls6qRNJqbz7CDkMC4G0g4+tQ6NZvqHjC31GLWJdVt7G3eP7QyKELPj5VKgAkYyfwrtWUMuGAI9CKFRUGEUKPQDFNU7STQ/r0+WS6u/orqz3u9V5i0UUVqeeFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAH/9k=)

참고문헌 : [Second Chance (or Clock) Page Replacement Policy - GeeksforGeeks](https://www.geeksforgeeks.org/second-chance-or-clock-page-replacement-policy/)

* + ESC

페이지 read 비트와 페이지 write 비트를 고려하여 페이지를 교체한다. circular queue로 구현했으며 pointer는 가장 오래된 page를 가리킨다. Read bit와 write bit는 page 랜덤으로 결정된 값(0 또는 1)으로 각각 설정된다. 단, 처음 프레임에 들어온 page의 read bit는 기본적으로 1로 설정된다. 그리고 이미 프레임에 페이지가 존재하는 경우 원래 존재하던 페이지의 r/w bit가 0이면 1로 수정되지만, 1인 경우에는 0으로 수정되지 않는다. 즉, 기존의 r/w bit가 (1,1)이고, 새로 들어온 r/w bit가 (1,0)이면 (1,1)상태를 유지한다. 하지만 기존의 r/w bit가 (0,0)이고, 새로 들어온 r/w bit가 (0,1)이면 (0,1)상태로 수정한다. page fault가 발생하면 현재 포인터 위치에서 프레임 버퍼들을 scan하면서 (0,0)상태를 가진 첫 페이지를 만나면 해당 페이지를 교체한 후 포인터는 다음 페이지를 가리킨다. (0,0) 상태를 가진 페이지를 만나지 못했다면 (0,1) 상태의 프레임을 찾아 다시 스캔한다. 이번 스캔 과정에서는 지나쳐가는 모든 프레임의 read bit를 0으로 설정한다. 프레임 버퍼들을 scan하면서 (0,1) 상태를 가진 페이지를 만나면 해당 페이지를 교체한 후 포인터는 다음 페이지를 가리킨다. (0,1) 상태를 가진 프레임을 만나지 못했다면 포인터는 원래의 시작 위치로 돌아와 다시 (0,0)상태를 가진 페이지부터 스캔하는 과정을 반복한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

참고문헌 : 김철홍 교수님 운영체제(라) 강의 자료 및 구두 설명

* 구현한 페이지 교체 알고리즘의 Pseudo code
  + Optimal

for i = 0 to pageStringNum do

    if page hit then

         do nothing.

    else if page miss and no swapping then

         load page into empty frame.

         pagefault++

    else if page miss and swapping then

         for j=0 to pageframesNum do

            search future index of pages that will be referenced.

         endfor

         for j=0 to pageframesNum do

            find a farthest page that is referenced in future.

         endfor

         page replacement

         pagefault++

    endif

    print frames state

endfor

* + FIFO

for i = 0 to pageStringNum do

    if page hit then

         do nothing.

    else if page miss and no swapping then

         load page into empty frame.

         pagefault++

    else if page miss and swapping then

         page replacement between oldest page in queue and incomming.

         pagefault++

    endif

    print frames state

endfor

* + LIFO

for i = 0 to pageStringNum do

    if page hit then

         do nothing.

    else if page miss and no swapping then

         load page into stack[top++].

         pagefault++

    else if page miss and swapping then

         load page into stack[top].

         pagefault++

    endif

    print frames state

endfor

* + LRU

for i = 0 to pageStringNum do

    if page hit then

         record time

         timer++

    else if page miss and no swapping then

         load page into empty frame

         record time

         timer++

         pagefault++

    else if page miss and swapping then

         for j=0 to pageframesNum do

            search oldest used page and page#

            select a page that is oldest used.

         endfor

         page replacement between oldest used page and incomming.

         record time.

         timer++

         pagefault++

    endif

    print frames state

endfor

* + LFU

for i = 0 to pageStringNum do

    if page hit then

         frequency++

    else if page miss and no swapping then

         load page into empty frame

         frequency=0

         pagefault++

    else if page miss and swapping then

         for j=0 to pageframesNum do

            search least frequency page and page#

         endfor

         page replacement between least frequency page and incomming.

         frequency=0

         pagefault++

    endif

    print frames state

endfor

* + SC

for i = 0 to pageStringNum do

    if page hit then

         second\_chance=1

    else if page miss then

         while true

             if second\_chance==0 then

                page replacement

                pointer=next page

                break

             second\_chance=0

         endwhile

         pagefault++

    endif

    print frames state

endfor

* + ESC

for i = 0 to pageStringNum do

     if page hit then

         modify read bit

         modify write bit

     endif

    if page miss and no swapping then

         load page into empty frame

         set read bit =1

         set write bit

         pagefault++

         set queue pointer to next page

     endif

    loop :

    if page miss and swapping then

         for j=0 to pageframesNum do

               if pointer meet (0,0) then

                    pae replacement between (0,0) and incomming

                    set read bit =1

                    pagefault++

                    set queue pointer to next page

                    break

               endif

               set queue pointer to next page

          endfor

    endif

    if page miss and swapping and not exist (0,0) then

         for j=0 to pageframesNum do

               if pointer meet (0,1) then

                    pae replacement between (0,1) and incomming

                    set read bit =1

                    pagefault++

                    set queue pointer to next page

                    break

               else

                    set read bit =0

                    set queue pointer to next page

               endif

               set queue pointer to next page

          endfor

    endif

    if page miss and swapping and not exist (0,0) and not exist (0,1) then

         goto loop

    endif

    print frames state

endfor

* 구현한 스케쥴링 알고리즘의 성능 분석 비교
* 주석이 포함된 소스코드
  + Optimal

void optimal1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

     printf(">>>>>>>>>>>>>>>>>>>>Optimal<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>Optimal<<<<<<<<<<<<<<<<<\n");

    int pageStringNum=REFERENCE\_STRING\_NUM;

    int frames[pageframesNum], temp[pageframesNum];

    int check1, check2, check3;

    int pos, max;

    int pageFaults = 0;

    char c='?';

    for(int i = 0; i < pageframesNum; ++i){

        frames[i] = -1;

    }

    for(int i = 0; i < pageStringNum; ++i){

        check1 = 0;

        check2 = 0;

        for(int j = 0; j < pageframesNum; ++j){

            if(frames[j] == buf[i]){    //page hit

                   check1 = 1;

                   check2 = 1;

                   c='H';

                   break;

               }

        }

        if(check1 == 0){ //page miss & no swapping(just insert page)

            for(int j = 0; j < pageframesNum; ++j){

                if(frames[j] == -1){    //no swap

                    pageFaults++;

                    frames[j] = buf[i]; //insert page

                    check2 = 1;

                    c='F';

                    break;

                }

            }

        }

        if(check2 == 0){ //page miss & swapping

            check3 =0;

            for(int j = 0; j < pageframesNum; ++j){

                temp[j] = -1;

                for(int k = i + 1; k < pageStringNum; ++k){

                    if(frames[j] == buf[k]){

                        temp[j] = k;    //현재 프레임에 저장된 string이 이후에 등장하명 해당 index 저장, 아니면 -1.

                        break;

                    }

                }

            }

            for(int j = 0; j < pageframesNum; ++j){

                if(temp[j] == -1){  //optimal, 현재 프레임에 저장된 string 중 이후에 나오지 않는 string 존재.

                    pos = j;

                    check3 = 1;

                    break;

                }

            }

            if(check3 ==0){  //optimal, 현재 프레임에 저장된 string이 모두 이후에 등장함.

             max = temp[0];

             pos = 0;

                for(int j = 1; j < pageframesNum; ++j){

                    if(temp[j] > max){  //optimal, 현재 프레임에 저장된 string 중 가장 나중에 등장하는 string 찾기.

                    max = temp[j];

                    pos = j;

                    }

                }

            }

            frames[pos] = buf[i];   //가장 나중에 사용될 string 을 page replacement

            pageFaults++;   //count page fault

            c='F';

        }

        //print state

        printf("%d\t\t\t",buf[i]);

        fprintf(fp, "%d\t\t\t", buf[i]);

        for(int j = 0; j < pageframesNum; ++j){

            if(frames[j] != -1){

                printf(" %d\t\t\t", frames[j]);

                fprintf(fp, "%d\t\t\t", frames[j]);

            }else{

                printf(" - \t\t\t");

                fprintf(fp, " - \t\t\t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("Optimal Total Page Faults:\t%d\n", pageFaults);

    fprintf(fp, "Optimal Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

* + FIFO

void fifo1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>FIFO<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>FIFO<<<<<<<<<<<<<<<<<\n");

    int pageFaults = 0;

    int check;

    int pageStringNum= REFERENCE\_STRING\_NUM;

    char c='?';

    int frame[pageframesNum];

    for(int m = 0; m < pageframesNum; m++)

    {

        frame[m] = -1;

    }

    for(int m = 0; m < pageStringNum; m++)

    {

        check = 0;

        for(int n = 0; n < pageframesNum; n++)

        {

            if(buf[m] == frame[n])  //psge hit

            {

                check=1;

                c='H';

            }

        }

        if((pageFaults < pageframesNum) && (check == 0)){   //page fault&no swapping(just insert page)

            pageFaults++;   //count page fault

            frame[(pageFaults - 1) % pageframesNum] = buf[m];  //insert page

            c='F';

        }else if(check == 0) {  //fault&swapping

            pageFaults++;   //count page fault

            frame[(pageFaults - 1) % pageframesNum] = buf[m];   //page replacement

            c='F';

        }

        //print state

        printf("%2d\t",buf[m]);

        fprintf(fp, "%2d\t", buf[m]);

        for(int n = 0; n < pageframesNum; n++)

        {

            if(frame[n] != -1){

                printf(" %2d\t", frame[n]);

                fprintf(fp, "%2d\t", frame[n]);

            }else{

                printf(" - \t");

                fprintf(fp, " - \t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("FIFO Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "FIFO Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

* + LIFO

void lifo1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>LIFO<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>LIFO<<<<<<<<<<<<<<<<<\n");

    int pageFaults = 0;

    int check;

    int pageStringNum=REFERENCE\_STRING\_NUM;

    char c='?';

    int stack[pageframesNum];   //create stack

    int top=-1;

    for(int m = 0; m < pageframesNum; m++)

    {

        stack[m] = -1;

    }

    for(int m = 0; m < pageStringNum; m++)

    {

        check=0;

        for(int n = 0; n < pageframesNum; n++) //PAGE FAULT 검사

        {

            if(buf[m] == stack[n])    //PAGE HIT

            {

                check=1;

                c='H';

            }

        }

        //FAULT인 경우 올바른 FRAME 위치에 저장

        if((pageFaults < pageframesNum) && (check == 0))  //PAGE FAULT&&NO SWAPPING

        {

            stack[++top]= buf[m];   //INSERT PAGE INTO STACK

            pageFaults++;   //COUNT PAGE FAULT

            c='F';

        }

        else if(check == 0)     //PAGE FAULT&SWAPPING

        {

            stack[top]= buf[m];     //INSERT PAGE INTO TOP OF STACK

            pageFaults++;   //COUNT PAGE FAULT

            c='F';

        }

        //PRINT STATE

        printf("%d\t\t\t",buf[m]);

        fprintf(fp, "%d\t\t\t", buf[m]);

        for(int n = 0; n < pageframesNum; n++)

        {

            if(stack[n] != -1){

                printf(" %d\t\t\t", stack[n]);

                fprintf(fp, "%d\t\t\t", stack[n]);

            }else{

                printf(" - \t\t\t");

                fprintf(fp, " - \t\t\t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("LIFO Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "LIFO Total Page Faults : %d\n", pageFaults);

    fclose(fp);

}

* + LRU

void lru1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>LRU<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>LRU<<<<<<<<<<<<<<<<<\n");

    int pageStringNum=REFERENCE\_STRING\_NUM;

    int frames[pageframesNum];

    int temp[pageframesNum];

    int counter = 0;

    int check1, check2, pageFaults = 0;

    char c='?';

    for(int i = 0; i < pageframesNum; ++i){  //initialize

        frames[i] = -1;

    }

    for(int i = 0; i < pageStringNum; ++i){

        check1 = 0;

        check2 = 0;

        for(int j = 0; j < pageframesNum; ++j){

            if(frames[j] == buf[i]){  //page hit

                counter++;  //increase time(=counter)

                temp[j] = counter;  //record time(=counter)

                check1 = 1;

                check2 = 1;

                c='H';

                break;

            }

        }

        if(check1 == 0){     //page fault & no swapping(just insert page)

            for(int j = 0; j < pageframesNum; ++j){

                if(frames[j] == -1){    //empty page

                    counter++;  //increase time(=counter)

                    pageFaults++;   //count page fault

                    frames[j] = buf[i];   //page replacement

                    temp[j] = counter;  //record time(=counter)

                    check2 = 1;

                    c='F';

                    break;

                }

            }

        }

        if(check2 == 0){     //page fault & swapping

            int minimum = temp[0];

            int pos=0;

            for(int j = 1; j < pageframesNum; ++j){

                if(temp[j] < minimum){  //select old used page

                    minimum = temp[j];

                    pos = j;    //old used page's index

                }

            }

            counter++;  //increase time(=counter)

            pageFaults++;   //count page fault

            frames[pos] = buf[i]; //page replacement

            temp[pos] = counter;    //record time(=counter)

            c='F';

        }

        //print state

        printf("%d\t\t\t",buf[i]);  //print incomming page string

        fprintf(fp, "%d\t\t\t", buf[i]);

        for(int j = 0; j < pageframesNum; ++j){  //print pages state

            if(frames[j] != -1){

                printf(" %d\t\t\t", frames[j]);

                fprintf(fp, "%d\t\t\t", frames[j]);

            }else{

                printf(" - \t\t\t");

                fprintf(fp, " - \t\t\t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("LRU Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "LRU Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

* + LFU

void lfu1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>LFU<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>LFU<<<<<<<<<<<<<<<<<\n");

        int frequency[pageframesNum];   //counter

        memset(frequency, 0, pageframesNum\*sizeof(frequency[0]));

        int frames[pageframesNum];

        memset(frames, -1, pageframesNum\*sizeof(frames[0]));

        int len=REFERENCE\_STRING\_NUM;

        int pageFaults=0;

        char c='?';

        for(int i=0; i< len; i++){

            int check = 0;

            int min,minpos;

            for(int j=0; j<pageframesNum; j++){

                if (frames[j] == buf[i]){   //page hit

                    frequency[j]++; //increase frequency(counter)

                    check = 1;

                    c='H';

                    break;

                }

            }

            if(check == 0){ //page miss

                for(int j=0; j<pageframesNum; j++){

                    if(frames[j] == -1){    //no swapping. just insert page.

                        frames[j] = buf[i];

                        frequency[j];   //not increase frequency

                        check = 1;

                        pageFaults++;   //count page fault

                        c='F';

                        break;

                    }

                }

            }

            if(check == 0){ //page miss & swapping

                min=frequency[0];

                minpos=0;

                for(int k = 1; k< pageframesNum; k++){

                    if(frequency[k] < min){ //select least frequency page

                        min = frequency[k];

                        minpos = k;

                    }

                }

                frames[minpos] = buf[i];    //page swapping

                frequency[minpos] = 0;  //set frequency(counter)

                pageFaults++;   //count pagefault

                c='F';

            }

            //print state

            printf("%d\t\t\t",buf[i]);  //print incomming page string

            fprintf(fp, "%d\t\t\t", buf[i]);

            for(int m = 0; m< pageframesNum; m++){

                if(frames[m] != -1){

                    printf(" %d(%d)\t\t\t", frames[m],frequency[m]);

                    fprintf(fp, "%d(%d)\t\t\t", frames[m],frequency[m]);

                }else{

                    printf(" -(-) \t\t\t");

                    fprintf(fp, " -(-) \t\t\t");

                }

            }

            printf("%c\n",c);

            fprintf(fp, "%c\n",c);

        }

    printf("LFU Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "LFU Total Page Faults : %d\n", pageFaults);

    fclose(fp);

}

* + SC

void sc1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>SC<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>SC<<<<<<<<<<<<<<<<<\n");

    int pointer=0;

    int pageFaults = 0;

    int pageStringNum=0;

    char c='?';

    int frame[pageframesNum];   //frame

    memset(frame, -1, sizeof(frame[0])\*pageframesNum);

    int second\_chance[pageframesNum];   //R bit

    memset(second\_chance, 0, sizeof(second\_chance[0])\*pageframesNum);

    pageStringNum=REFERENCE\_STRING\_NUM;

    for(int i = 0; i < pageStringNum; i++)

    {

        int check=0;

        for(int j = 0; j < pageframesNum; j++)

        {

            if(frame[j] == buf[i])  //page hit

            {

                second\_chance[j] = 1;   //set second chance bit = 1

                check=1;

                c='H';

            }

        }

        if(!check)  //page miss

        {

            while(1)

            {

                if(!second\_chance[pointer]) //if secod chance==0

                {

                    frame[pointer] = buf[i];        //page replacement

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    break;

                }

                second\_chance[pointer] = 0; //if second chance == 1 then second chance=0

                pointer = (pointer + 1) % pageframesNum;    //set queue pointer to next page

            }

            pageFaults++;   //count pagefault

            c='F';

        }

        //print state

        printf("%d\t\t\t",buf[i]);  //print incomming page string

        fprintf(fp, "%d\t\t\t", buf[i]);

        for(int m = 0; m< pageframesNum; m++){

            // printf("%d(%d)\t",frame[m],second\_chance[m]);

            if(frame[m] != -1){

                printf(" %d(%d)\t\t\t", frame[m],second\_chance[m]);

                fprintf(fp, "%d(%d)\t\t\t", frame[m],second\_chance[m]);

            }else{

                printf(" -(-) \t\t\t");

                fprintf(fp, " -(-) \t\t\t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("SC Total Page Faults : %d\n",pageFaults);

    fprintf(fp, "SC Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

* + ESC

void esc1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>ESC<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>ESC<<<<<<<<<<<<<<<<<\n");

    int pointer=0;

    int pageFaults = 0;

    int check1,check2,check3,check4;

    int pageStringNum=REFERENCE\_STRING\_NUM;

    char c='?';

    int frame[pageframesNum];   //frame

    memset(frame, -1, sizeof(frame[0])\*pageframesNum);

    int r[pageframesNum];   //R bit

    memset(r, 1, sizeof(r[0])\*pageframesNum);

    int w[pageframesNum];   //W bit

    memset(w, 0, sizeof(w[0])\*pageframesNum);

    pageStringNum=REFERENCE\_STRING\_NUM;

    for(int i = 0; i < pageStringNum; i++)

    {

        check1=check2=check3=check4=0;

        for(int j = 0; j < pageframesNum; j++)

        {

            if(frame[j] == buf[i])  //page hit

            {

                r[j] = read[i]==1?1:r[j];   //set read bit

                w[j] = write[i]==1?1:w[j];     //set write bit

                check1=check2=check3=check4=1;

                c='H';

            }

        }

        if(!check1)  //page miss & no swapping

        {

            for(int j = 0; j < pageframesNum; ++j){

                if(frame[j] == -1){    //empty page

                    frame[j] = buf[i];   //just load page

                    r[j]=read[i];   //set r bit

                    w[j]=write[i];  //set w bit

                    r[j]=1;

                    check2 =check3=check4= 1;

                    pageFaults++;   //count pagefault

                    c='F';

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    break;

                }

            }

        }

        //goto

        LOOP:

        if(check2==0){  //page fault & swapping between (0,0) and incomming

            for(int j = 0; j < pageframesNum; ++j){

                if(((r[pointer]==0)&&(w[pointer]==0))) //if (0,0)

                {

                    frame[pointer] = buf[i];        //page replacement

                    r[pointer]=read[i];

                    w[pointer]=write[i];

                    r[pointer]=1;

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    check3=check4= 1;

                    pageFaults++;   //count pagefault

                    c='F';

                    break;

                }

                // r[pointer] = 0; //if r bit == 1 then r bit=0

                // w[pointer] = 0; //if w bit == 1 then w bit=0

                pointer = (pointer + 1) % pageframesNum;    //set queue pointer to next page

            }

        }

        if(check3==0){  //page fault & swapping between (0,1) and incomming while (1,0),(1,1)->(0,0),(0,1)

            for(int j = 0; j < pageframesNum; ++j){

                if((((r[pointer]==0)&&(w[pointer]==1)))) //check (0,1) is exist

                {

                    frame[pointer] = buf[i];        //page replacement

                    r[pointer]=read[i];

                    w[pointer]=write[i];

                    r[pointer]=1;

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    check4=1;

                    pageFaults++;   //count pagefault

                    c='F';

                    break;

                }

                else{

                    r[pointer] = 0; //if r bit == 1 then r bit=0

                    pointer = (pointer + 1) % pageframesNum;    //set queue pointer to next page

                }

            }

        }

        if(check4==0){

            //check2,3반복

            goto LOOP;

        }

        //print state

        printf("%2d(%d,%d)\t\t",buf[i],read[i],write[i]);  //print incomming page string

        fprintf(fp, "%2d(%d,%d)\t\t", buf[i],read[i],write[i]);

        for(int m = 0; m< pageframesNum; m++){

            // printf("%d(%d)\t",frame[m],second\_chance[m]);

            if(frame[m] != -1){

                printf(" %2d(%d,%d)\t\t", frame[m],r[m],w[m]);

                fprintf(fp, "%2d(%d,%d)\t\t", frame[m],r[m],w[m]);

            }else{

                printf(" -(-,-)\t\t");

                fprintf(fp, " -(-,-)\t\t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("ESC Total Page Faults : %d\n",pageFaults);

    fprintf(fp, "ESC Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

* 사용자 메뉴얼
  + 알고리즘 시뮬레이터 선택. 단, ALL을 제외하고 최대 3개까지 가능.

텍스트, 키보드이(가) 표시된 사진

자동 생성된 설명

* + 페이지 프레임 개수 입력. 단, 최소 3에서 최대 10개까지 가능.

텍스트이(가) 표시된 사진

자동 생성된 설명

* + 데이터 입력 방식 (랜덤하게 생성(1), 사용자 생성 파일 오픈(2)) 선택.
    1. 랜덤하게 생성(1) 선택 시 프로그램 내부에서 rand() 함수를 사용하여 참조 스트링 생성 후 페이지 교체 알고리즘 실행.
    2. 사용자 생성 파일 오픈(2) 선택 시 파일명을 인자로 입력(1) 혹은 프로그램 내부에서 스트림 파일 생성(2) 추가 선택.
    3. 사용자 생성 파일 오픈(2) => 파일명을 인자로 입력(1) 선택 시 참조 스트링과 r/w bit가 이미 저장된 파일명을 입력하면, 프로그램 내부에서 해당 파일을 fopen(), fread()하여 페이지 교체 알고리즘 실행. 단, 해당 파일은 ‘string\tR/Wbit\n’ 형식으로 참조 스트링을 구분해야 함. 프로그램 내부에서 참조 스트링을 따로 생성해주지 않으므로 반드시 입력한 파일은 참조 스트링이 저장되어 있어야 함.
    4. 사용자 생성 파일 오픈(2) => 프로그램 내부에서 스트림 파일 생성(2) 선택 시 프로그램 내부에서 참조 스트링 및 파일(stream.txt)을 만든 후 이를 fopen(), fread()하여 페이지 교체 알고리즘 실행
    5. 시뮬레이션 결과는 표준 출력과 동시에 자동으로 파일(result.txt)로 저장

텍스트, 사슬이(가) 표시된 사진

자동 생성된 설명

Reference string

Frame0

Frame1

Frame2

Page fault/hit

그림 1 랜덤하게 생성(1) 시 시뮬레이션 결과 표준 출력 화면 일부 캡쳐

텍스트이(가) 표시된 사진

자동 생성된 설명

Reference string

Frame0

Frame1

Frame2

Page fault/hit

그림 2 랜덤하게 생성(1) 시 시뮬레이션 결과 파일 (result.txt) 자동 저장 화면 일부 캡쳐

텍스트, 저울, 사슬이(가) 표시된 사진

자동 생성된 설명

Reference string

Frame0

Frame1

Frame2

Page fault/hit

그림 3 사용자 생성 파일 오픈(2) => 파일명을 인자로 입력(1) 시 시뮬레이션 결과 표준 출력 화면 일부 캡쳐

텍스트이(가) 표시된 사진

자동 생성된 설명

Reference string

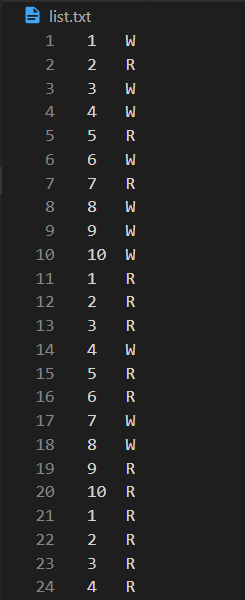
Frame0

Frame1

Frame2

Page fault/hit

그림 4 사용자 생성 파일 오픈(2) => 파일명을 인자로 입력(1) 시 시뮬레이션 결과 파일 (result.txt) 자동 저장 화면 일부 캡쳐



Reference string

R/W bit

그림 5 사용자 생성 파일 오픈(2) => 파일명을 인자로 입력(1) 선택 후 입력한 list.txt 파일 내부 캡쳐(‘string\tR/Wbit \n’ 형식으로 참조 스트링을 구분한 파일)

텍스트이(가) 표시된 사진

자동 생성된 설명

Reference string

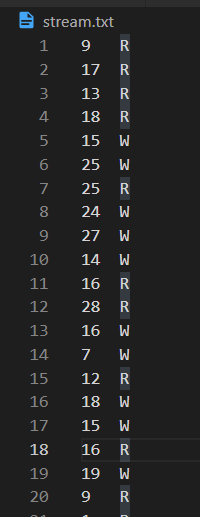
Frame0

Frame1

Frame2

Page fault/hit

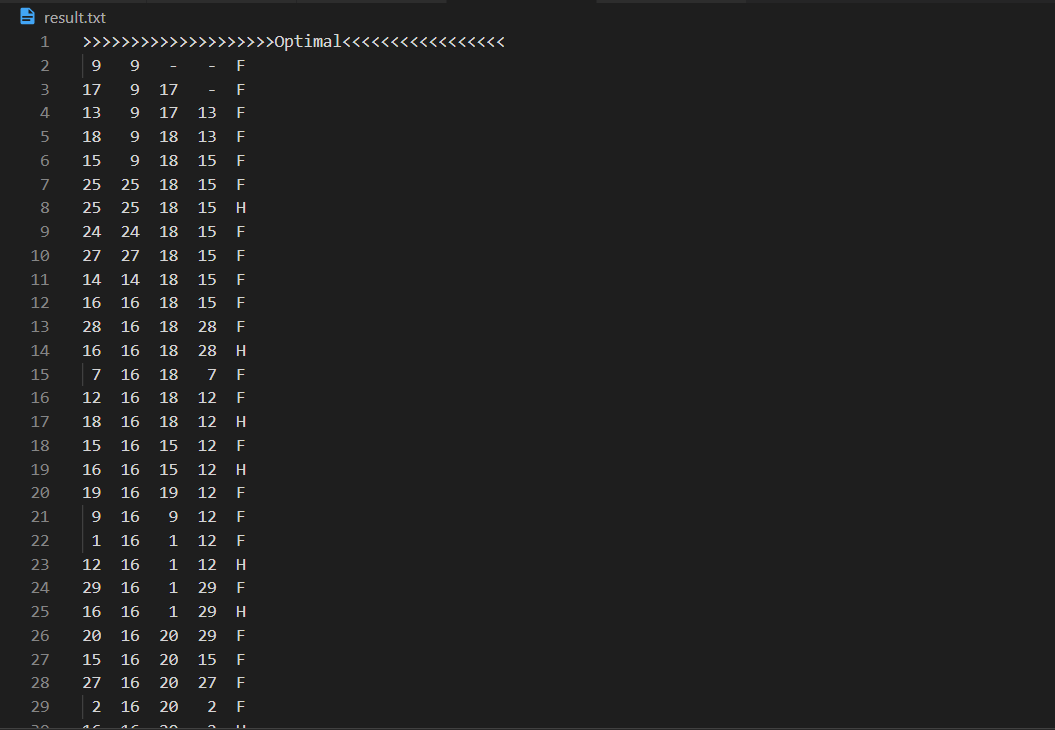
그림 6 사용자 생성 파일 오픈(2) => 프로그램 내부에서 스트림 파일 생성(2) 시 시뮬레이션 결과 표준 출력 화면 일부 캡쳐



Reference string

R/W bit

그림 7 사용자 생성 파일 오픈(2) => 프로그램 내부에서 스트림 파일 생성(2) 시 생성된 스트림 파일(stream.txt) 일부 캡쳐



Reference string

Frame0

Frame1

Frame2

Page fault/hit

그림 8 사용자 생성 파일 오픈(2) => 프로그램 내부에서 스트림 파일 생성(2) 시 시뮬레이션 결과 파일 (result.txt) 자동 저장 화면 일부 캡쳐

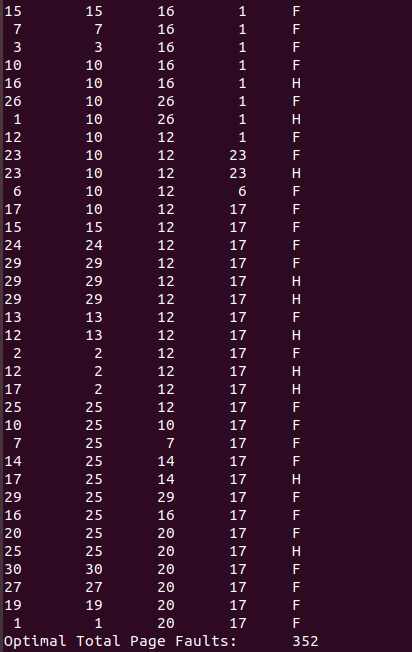
2. 결과

* 랜덤하게 생성
  + 표준 출력 결과

텍스트이(가) 표시된 사진

자동 생성된 설명

(((일부 생략)))



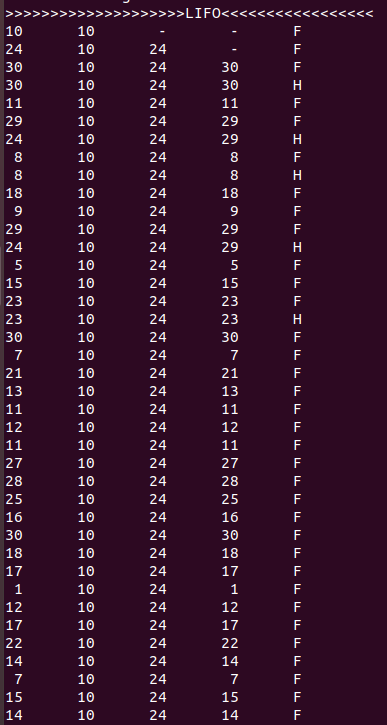
텍스트이(가) 표시된 사진

자동 생성된 설명

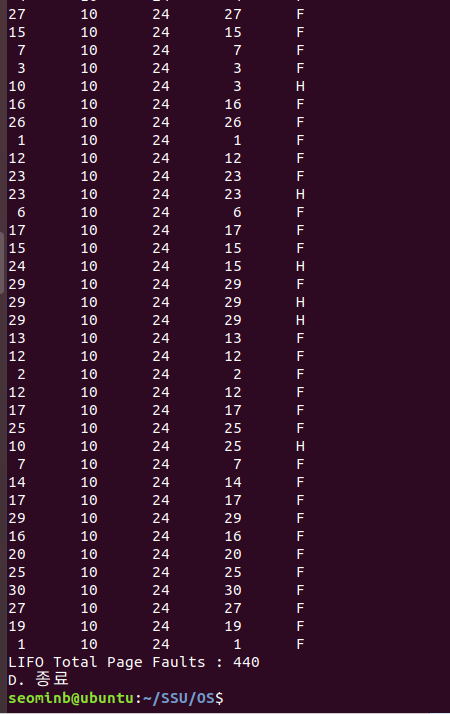
((일부생략))

텍스트이(가) 표시된 사진

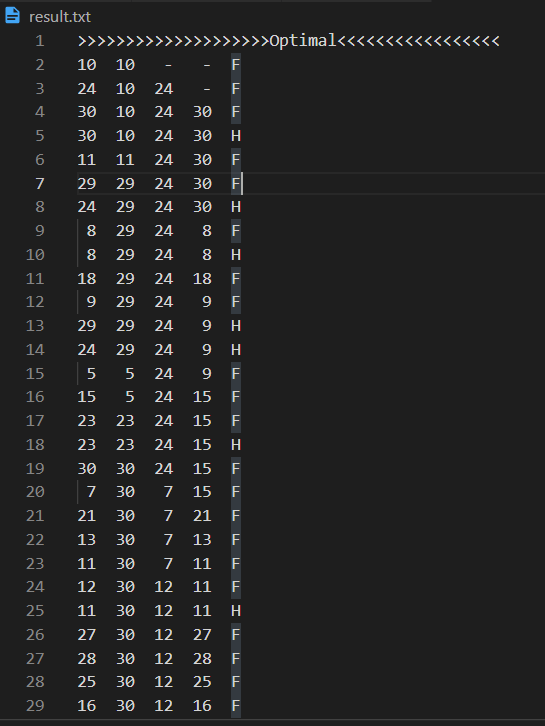
자동 생성된 설명



((일부생략))



* + 시뮬레이션 결과 자동 저장 파일 (result.txt)



((일부생략))

텍스트이(가) 표시된 사진

자동 생성된 설명

(((일부생략))

텍스트이(가) 표시된 사진

자동 생성된 설명

((일부생략))

텍스트, 전자기기, 키보드이(가) 표시된 사진

자동 생성된 설명

* 사용자 생성 파일 오픈 => 파일명을 인자로 입력
  + 인자로 입력한 파일(list.txt)

텍스트, 전자기기, 검은색, 키보드이(가) 표시된 사진

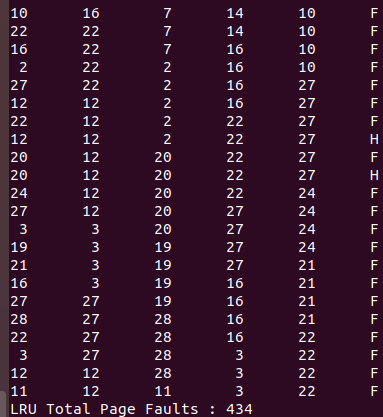
자동 생성된 설명

* + 표준 출력 결과

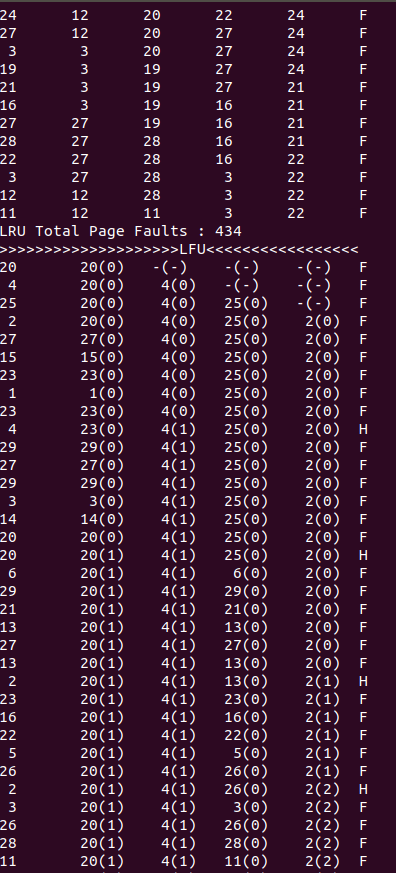
텍스트이(가) 표시된 사진

자동 생성된 설명

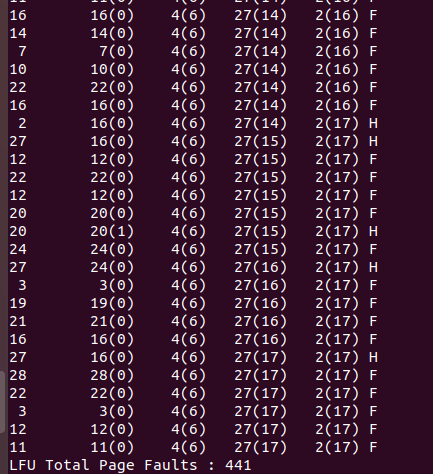
((일부생략))



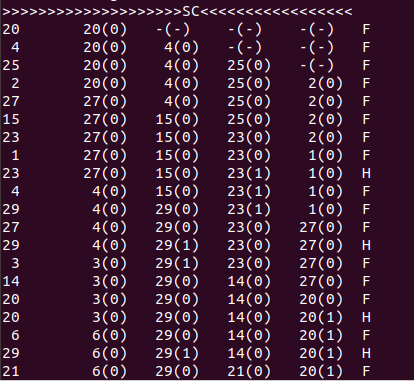
counter



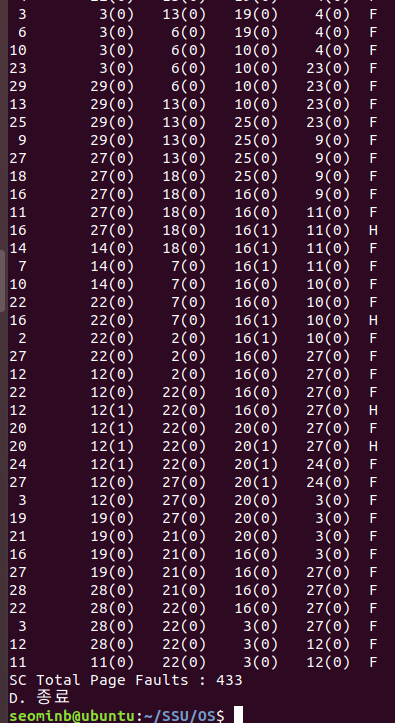
((일부생략)



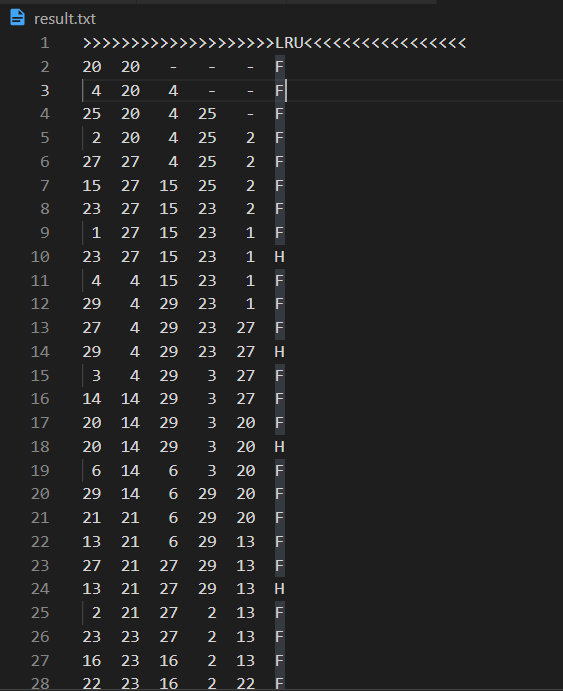
counter



((일부생략))



* + 시뮬레이션 결과 자동 저장 파일 (result.txt)



((일부생략))

텍스트이(가) 표시된 사진

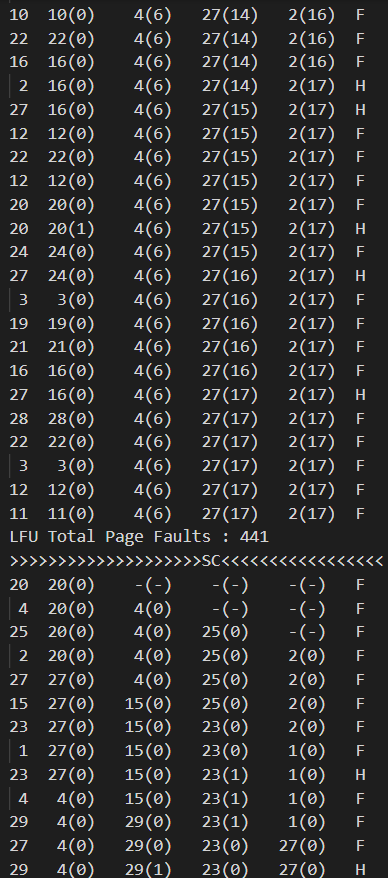
자동 생성된 설명

counter

텍스트이(가) 표시된 사진

자동 생성된 설명

((일부생략))



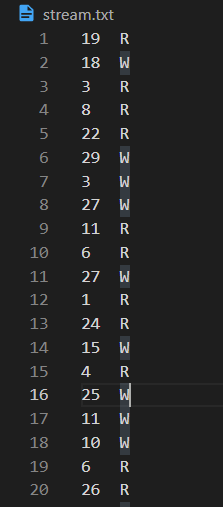
counter

((일부생략))

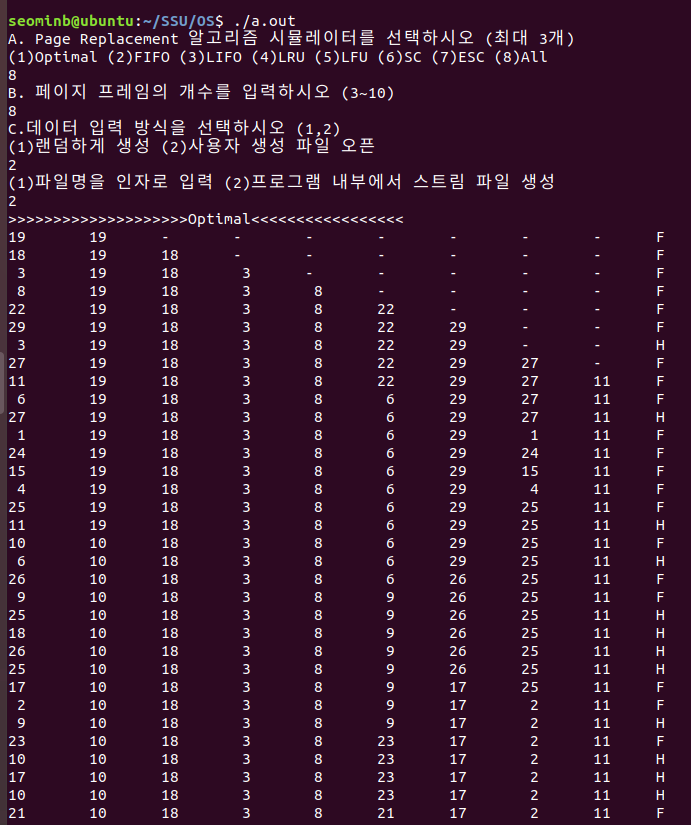
텍스트이(가) 표시된 사진

자동 생성된 설명

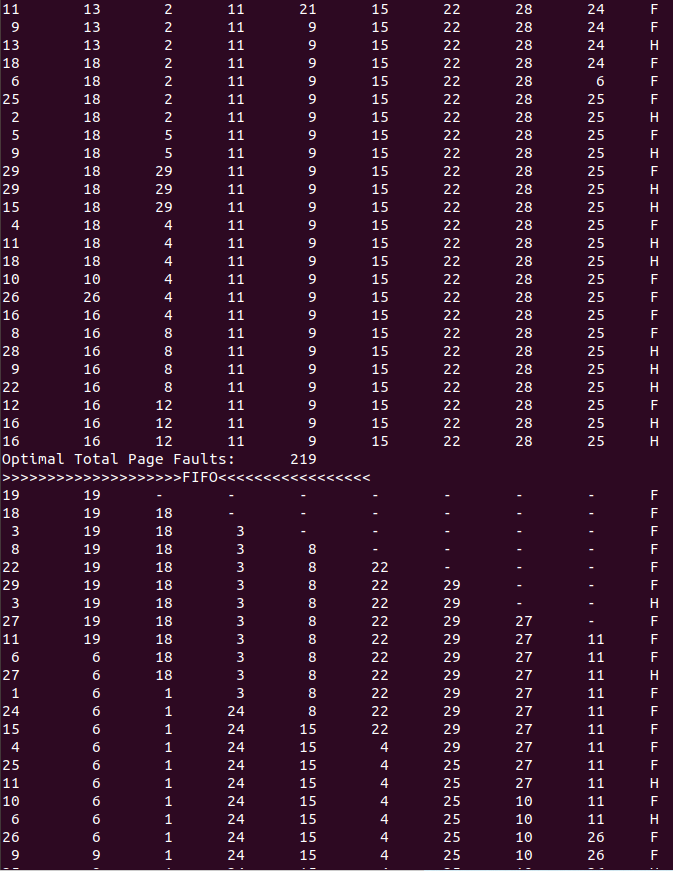
* 사용자 생성 파일 오픈 => (2)프로그램 내부에서 스트림 파일 생성
  + 스트림 파일(stream.txt) 생성



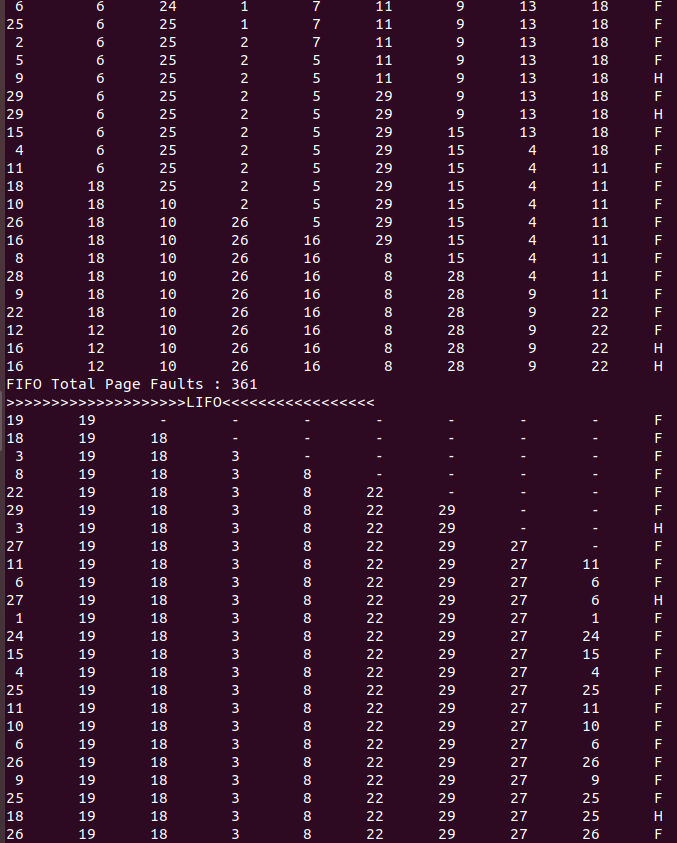
* + 표준 출력 결과



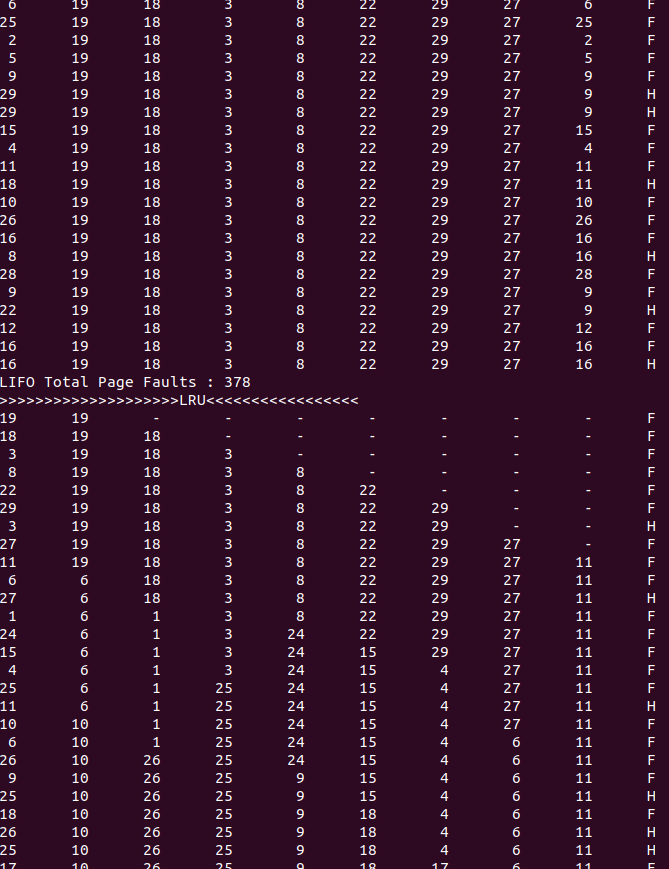
((일부생략))



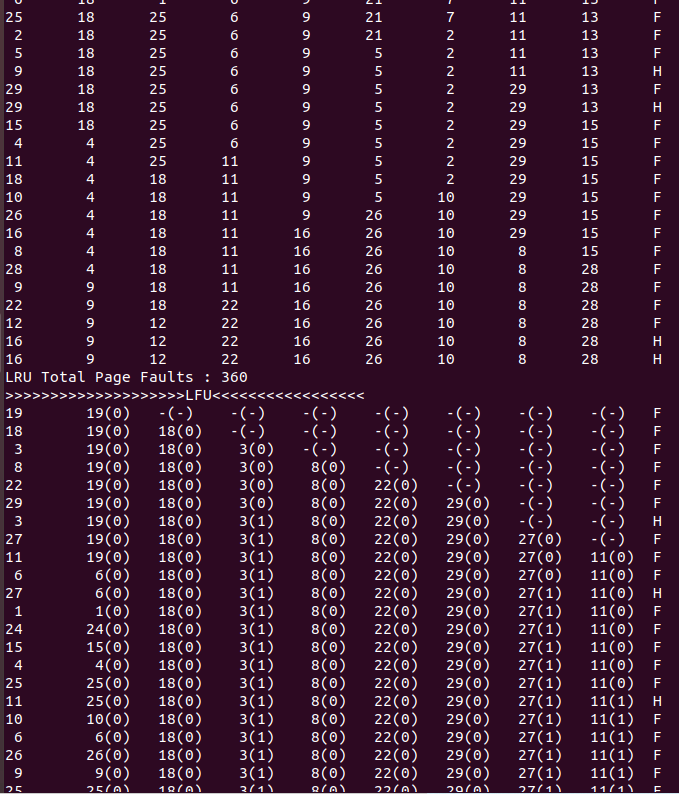
((일부생략))



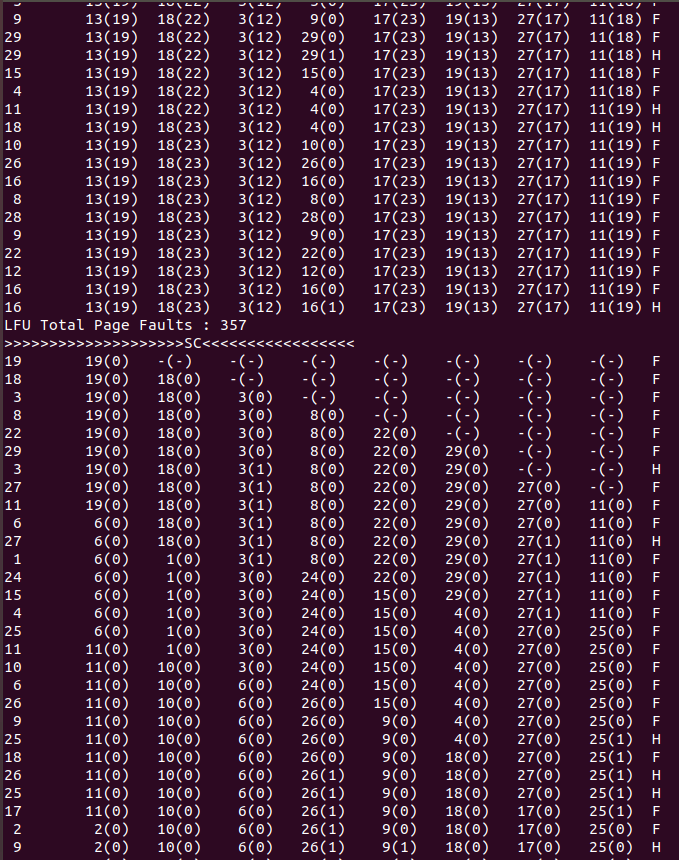
((일부생략))



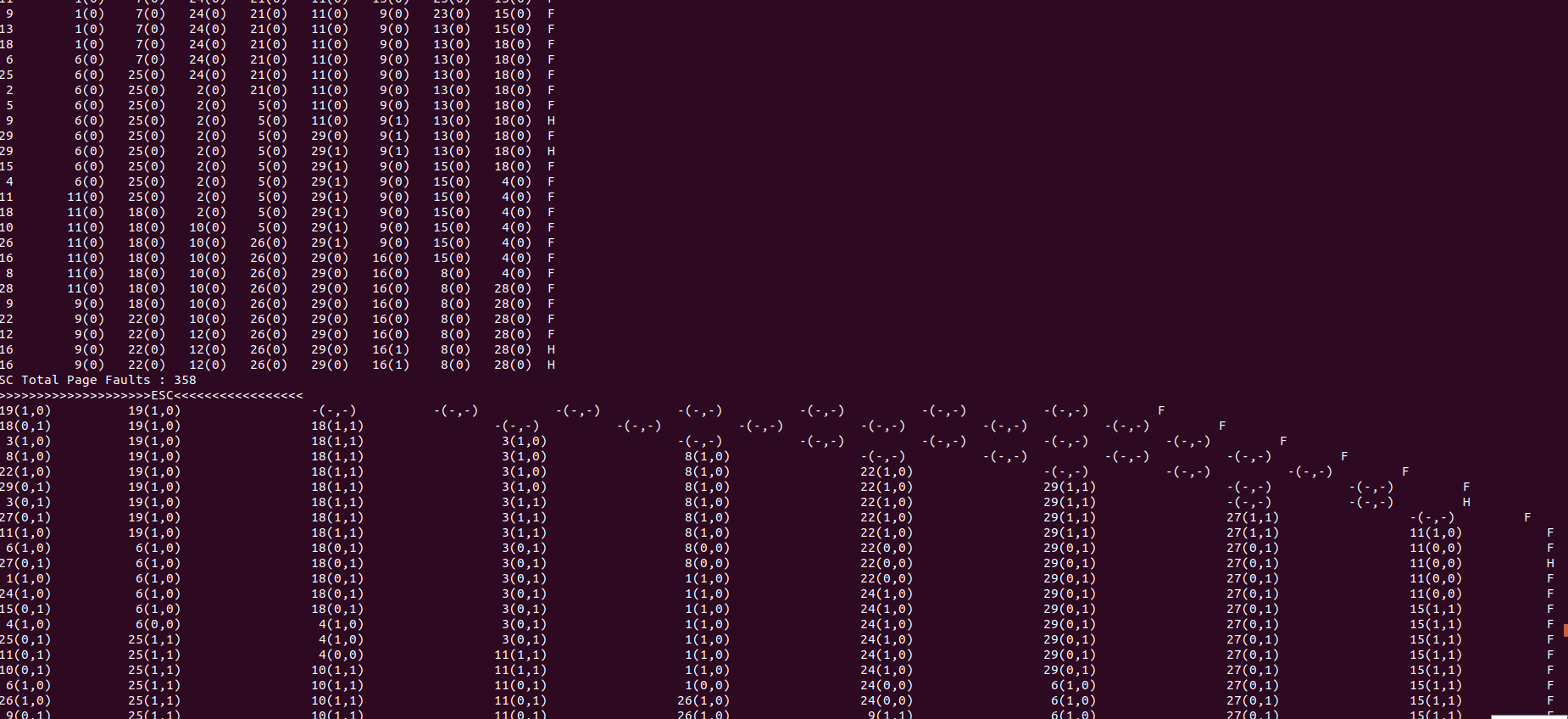
((일부생략))



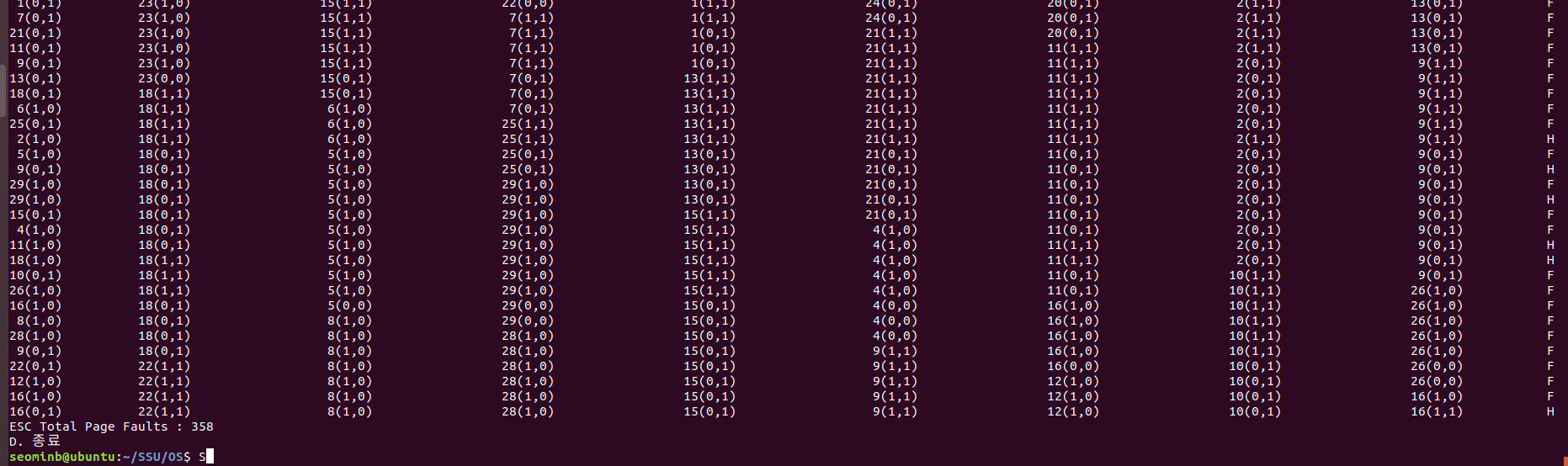
((일부생략))



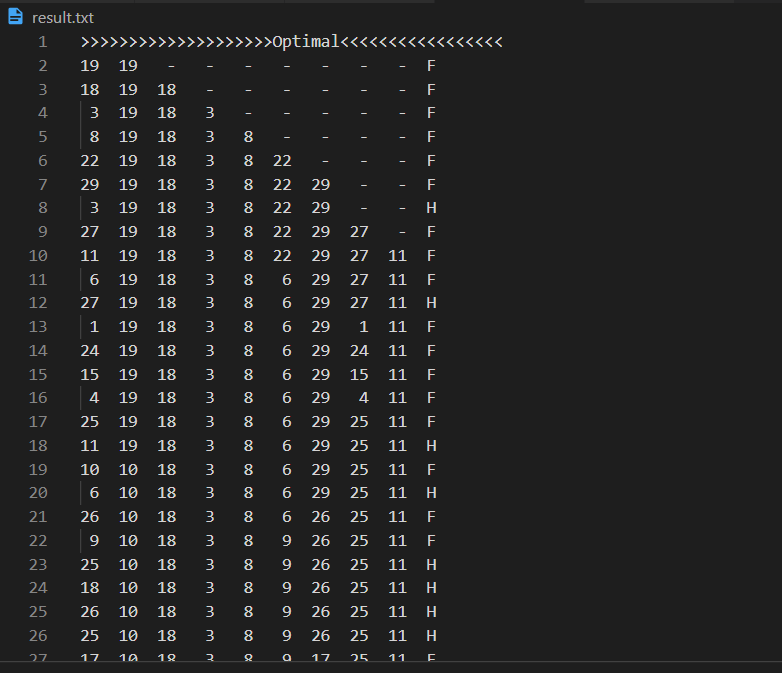
((일부생략))



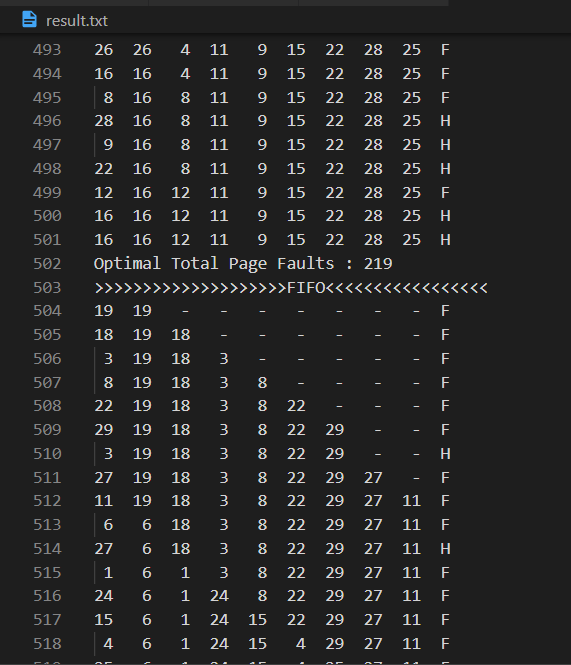
((일부생략))



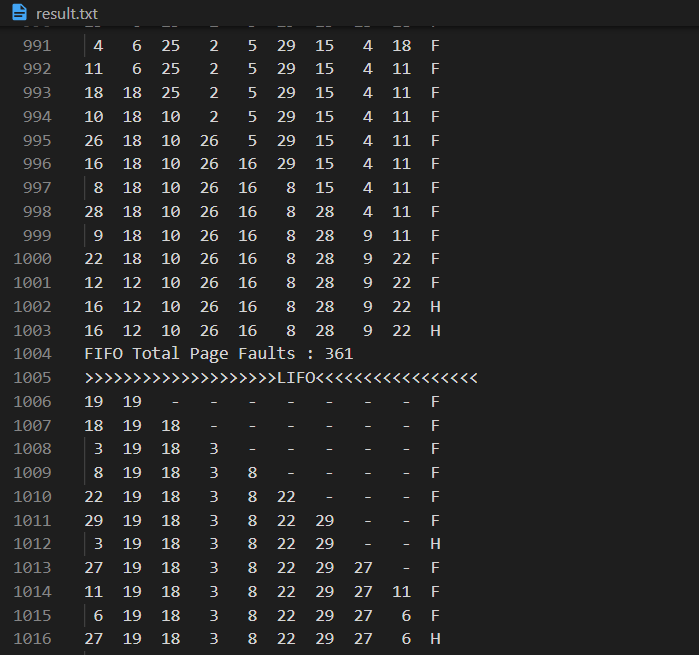
* + 시뮬레이션 결과 자동 저장 파일 (result.txt)



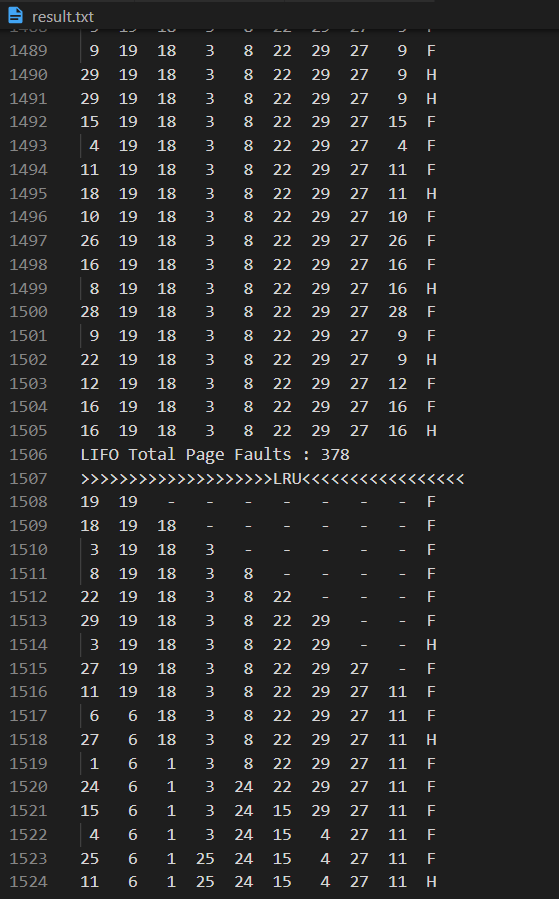
((일부생략))



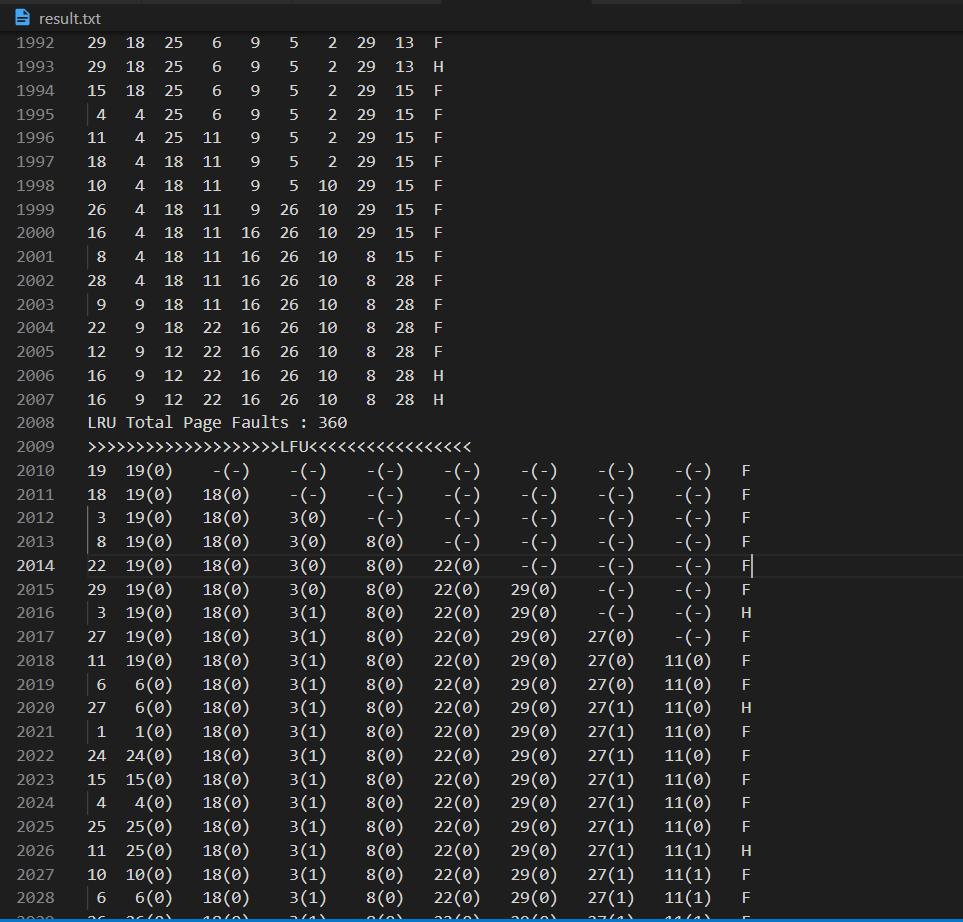
((일부생략))



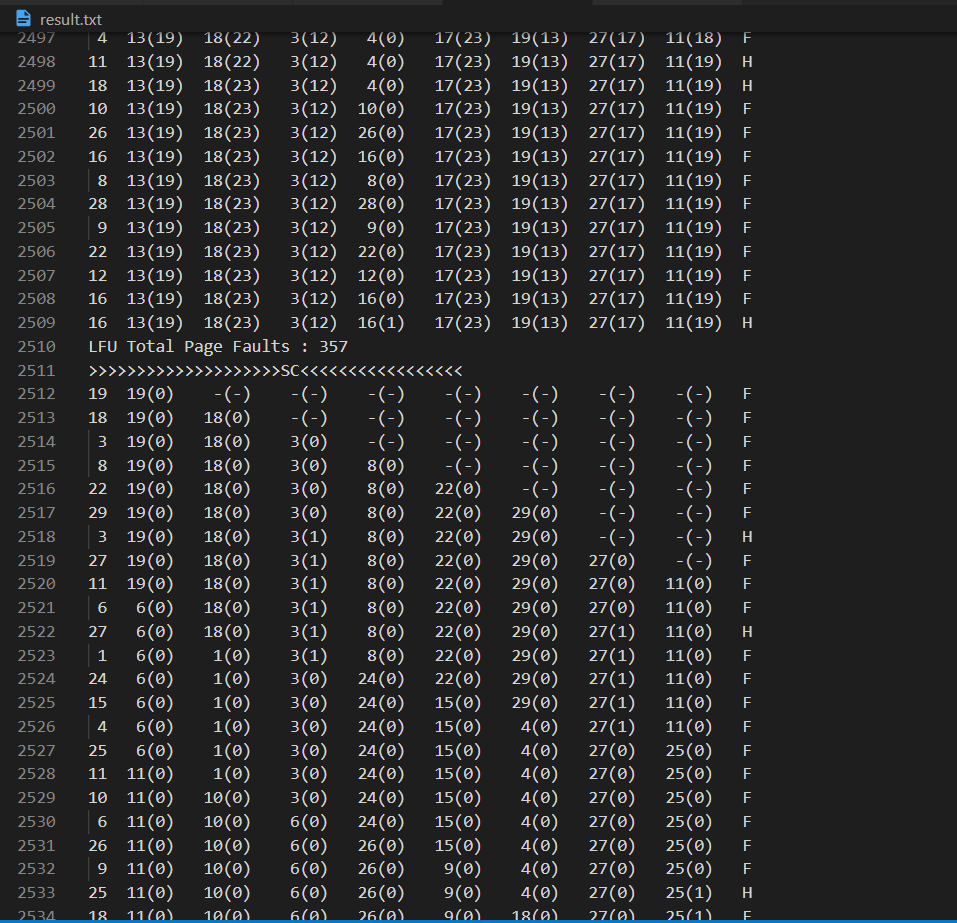
((일부생략))



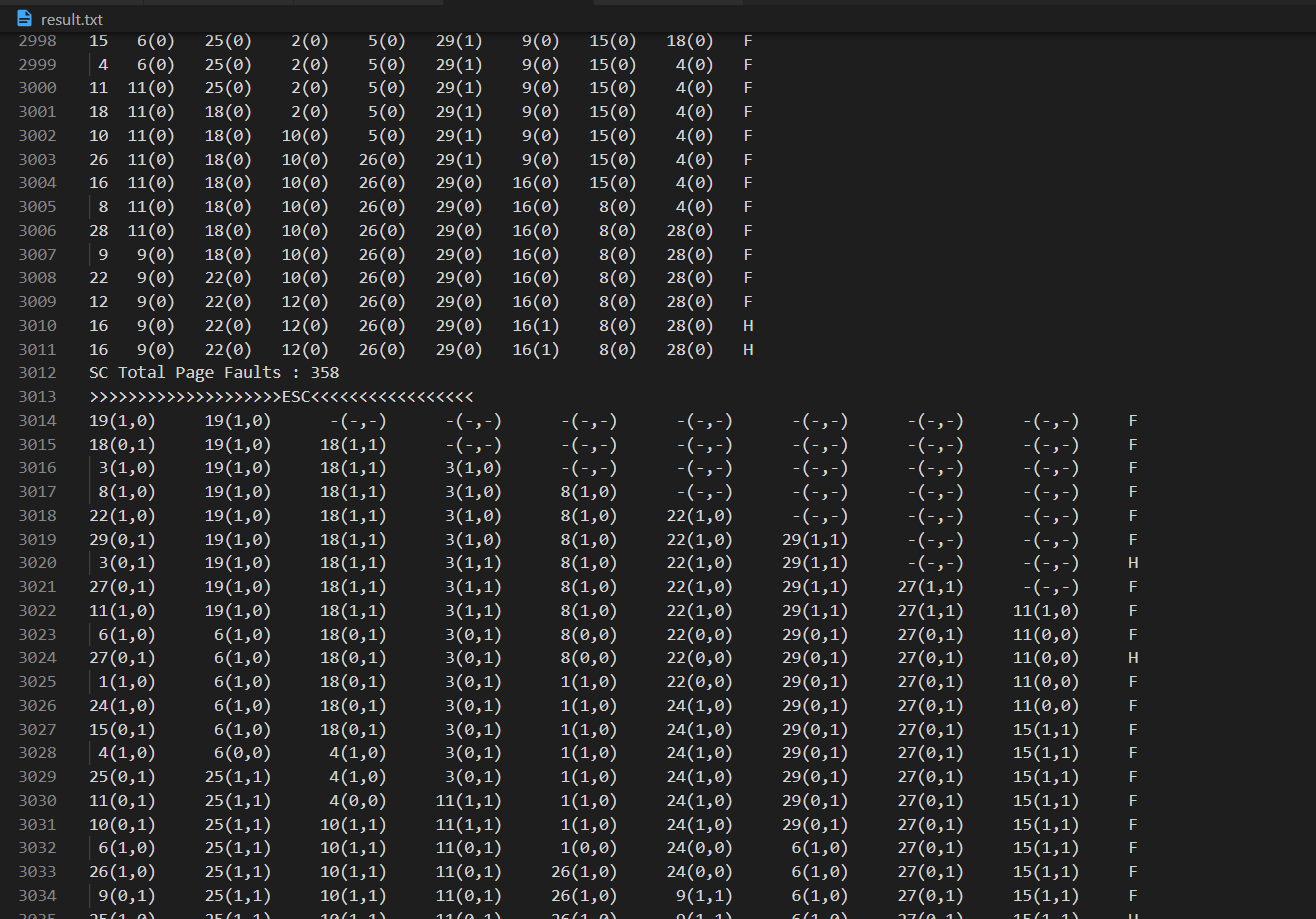
((일부생략))

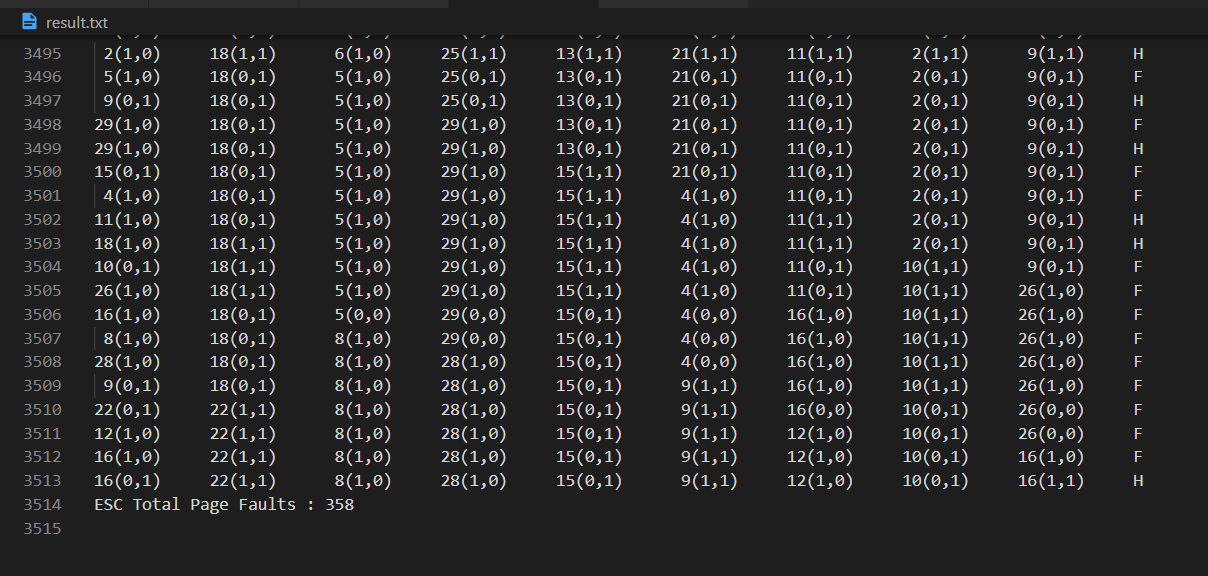


((일부생략))



((일부생략))





3. 소스코드

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <fcntl.h>

int \*buf;   //page stream

int \*write; //write bit for esc

int \*read;  //read bit for esc

int algo[8]={0,};   //select page replace algo

int pageframesNum; //페이지 프레임 개수

int data;   //데이터입력방식

int REFERENCE\_STRING\_NUM;    //참조 페이지 스트링 개수

void fifo1();

void lifo1();

void optimal1();

void lru1();

void lfu1();

void sc1();

void esc1();

void execAlgo();

void execAlgo(){

    if(algo[0]==1)

        optimal1();

    if(algo[1]==1)

        fifo1();

    if(algo[2]==1)

        lifo1();

    if(algo[3]==1)

        lru1();

    if(algo[4]==1)

        lfu1();

    if(algo[5]==1)

        sc1();

    if(algo[6]==1)

        esc1();

    if(algo[7]==1){

        optimal1();

        fifo1();

        lifo1();

        lru1();

        lfu1();

        sc1();

        esc1();

    }

    return;

}

int main(){

    printf("A. Page Replacement 알고리즘 시뮬레이터를 선택하시오 (최대 3개)\n");

    printf("(1)Optimal (2)FIFO (3)LIFO (4)LRU (5)LFU (6)SC (7)ESC (8)All\n");

    while(1){

        for(int j=0;j<8;j++){

            algo[j]=0;  //배열 초기화

        }

        char selectAlgo[100];

        scanf("%[^\n]s",selectAlgo);

        getchar();

        char \*ptr = strtok(selectAlgo, " ");      // " " 공백 문자를 기준으로 문자열을 자름, 포인터 반환

        int cnt=0;

        while (ptr != NULL)               // 자른 문자열이 나오지 않을 때까지 반복

        {

            cnt++;

            // printf("%d : %s %d\n",cnt, ptr, atoi(ptr));          // 자른 문자열 출력

            if(atoi(ptr)<=8 && atoi(ptr)>=1){

                algo[atoi(ptr)-1]=1;

            }else{

                cnt=-9999;

                break;

            }

            ptr = strtok(NULL, " ");      // 다음 문자열을 잘라서 포인터를 반환

        } //cnt=3개입력하면 3개로 종료.

        //all을 제외하고 최대 3개의 알고리즘 선택

        if(cnt<=3&&cnt>=1){

            if(algo[7]==1){

                if(cnt==1){

                    // printf("%d %d %d %d %d %d %d %d\n",algo[0],algo[1],algo[2],algo[3],algo[4],algo[5],algo[6],algo[7]);

                    break;

                }else{

                    // printf("%d %d %d %d %d %d %d %d\n",algo[0],algo[1],algo[2],algo[3],algo[4],algo[5],algo[6],algo[7]);

                    printf("입력 에러, ALL을 제외하고 최대 3개의 알고리즘(1~8)을 선택할 수 있습니다.\n");

                    continue;

                }

            }else{

                // printf("%d %d %d %d %d %d %d %d\n",algo[0],algo[1],algo[2],algo[3],algo[4],algo[5],algo[6],algo[7]);

                break;

            }

        }else{

            // printf("%d %d %d %d %d %d %d %d\n",algo[0],algo[1],algo[2],algo[3],algo[4],algo[5],algo[6],algo[7]);

            printf("입력 에러, ALL을 제외하고 최대 3개의 알고리즘(1~8)을 선택할 수 있습니다.\n");

            continue;

        }

    }

    printf("B. 페이지 프레임의 개수를 입력하시오 (3~10)\n");

    while(1){

        scanf("%d",&pageframesNum);

        // printf("pageframesNum : %d\n",pageframesNum);

        if(pageframesNum>=3 && pageframesNum<=10){    //입력 에러 처리

            break;

        }

        printf("입력 에러, 페이지 프레임 개수는 (3~10)입니다.\n");

    }

    printf("C.데이터 입력 방식을 선택하시오 (1,2)\n");

    printf("(1)랜덤하게 생성 (2)사용자 생성 파일 오픈\n");

    while(1){

        scanf("%d",&data);

        // printf("data : %d\n",data);

        if(data>=1 && data<=2){     //입력 에러 처리

            break;

        }

        printf("입력 에러, 데이터 입력 방식 선택은 (1,2)입니다.\n");

    }

    FILE \*fp;

    if((fp=fopen("result.txt","w+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    fclose(fp);

    switch(data){

        case 1:{

            REFERENCE\_STRING\_NUM=500;

            int rand\_reference\_string[REFERENCE\_STRING\_NUM];

            memset(rand\_reference\_string, 0, sizeof(rand\_reference\_string[0])\*REFERENCE\_STRING\_NUM);

            buf=rand\_reference\_string;

            int rand\_r[REFERENCE\_STRING\_NUM];   //Read bit

            memset(rand\_r, 0, sizeof(rand\_r[0])\*REFERENCE\_STRING\_NUM);

            read=rand\_r;

            int rand\_w[REFERENCE\_STRING\_NUM];   //write bit

            memset(rand\_w, 0, sizeof(rand\_w[0])\*REFERENCE\_STRING\_NUM);

            write=rand\_w;

            srand(time(NULL));  //랜덤하게 참조 스트링 생성.

            for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

                rand\_reference\_string[i]=(rand() % 30) + 1;

            // for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

            //     printf("%d ",buf[i]);

            // printf("\n");

            for (int i = 0; i < REFERENCE\_STRING\_NUM; i++){

                if(rand()%2==0){

                    rand\_r[i]=0;

                    rand\_w[i]=1;

                }

                else{

                    rand\_r[i]=1;

                    rand\_w[i]=0;

                }

            }

            // for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

            //     printf("%d ",rand\_r[i]);

            // printf("\n");

            // for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

            //     printf("%d ",rand\_w[i]);

            // printf("\n");

            execAlgo();

            // printf("case 1 : success\n");

            break;

        }

        case 2:{

            printf("(1)파일명을 인자로 입력 (2)프로그램 내부에서 스트림 파일 생성\n");

            int data2=0;

            while(1){

                scanf("%d",&data2);

                // printf("data2 : %d\n",data2);

                if(data2>=1 && data2<=2){     //입력 에러 처리

                    break;

                }

                printf("입력 에러, 파일 입력 방식 선택은 (1,2)입니다.\n");

            }

            char \*fname;

            if(data2==1){   //파일명 입력

                char filename[101];

                printf("파일명 : ");

                scanf("%s",filename);

                fname=malloc(strlen(filename) + 1);

                strcpy(fname, filename);

                fname[strlen(filename)] = '\0';

                // printf("파일명 : %s\n",fname);

            }else if(data2==2){ //프로그램 내부에서 파일 생성

                fname="stream.txt";

                FILE \*fp1;

                if((fp1=fopen(fname,"w+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

                    fprintf(stderr,"fopen error for %s\n",fname);

                    exit(1);

                }

                srand(time(NULL));  //랜덤하게 참조 스트링 생성.

                for (int i = 0; i < 500; i++){  //reference\_string\_num

                    fprintf(fp1, "%d\t", (rand() % 30) + 1);    //파일에 랜덤하게 생성한 참조 스트링을 저장.

                    fprintf(fp1, "%c\n", (rand() % 2)==0? 'R' : 'W');

                    // fprintf(fp1, "%c\n", (rand() % 2)==0? 'W' : '-');

                }

                fclose(fp1); //파일 닫기.

            }

            FILE \*fp2;

            if((fp2=fopen(fname,"r"))==NULL){ //읽기 전용으로 파일 오픈.

                fprintf(stderr,"fopen error for %s\n",fname);

                exit(1);

            }

            char \*line = NULL;

            size\_t size = 0;

            ssize\_t len;

            int cnt\_line=0; //파일 line 카운트.

            while ((len = getline(&line, &size, fp2)) != -1) {

                    // printf("Retrieved line of length %zu :\n", len);

                    // printf("%s", line);

                    cnt\_line++; //파일 line 수 카운트.

            }

            // printf("line : %d\n",cnt\_line);

            REFERENCE\_STRING\_NUM=cnt\_line;

            int file\_reference\_string[REFERENCE\_STRING\_NUM];

            memset(file\_reference\_string, 0, sizeof(file\_reference\_string[0])\*REFERENCE\_STRING\_NUM);

            buf=file\_reference\_string;

            int file\_r[REFERENCE\_STRING\_NUM];   //Read bit

            memset(file\_r, 0, sizeof(file\_r[0])\*REFERENCE\_STRING\_NUM);

            read=file\_r;

            int file\_w[REFERENCE\_STRING\_NUM];   //write bit

            memset(file\_w, 0, sizeof(file\_w[0])\*REFERENCE\_STRING\_NUM);

            write=file\_w;

            fseek(fp2, 0, SEEK\_SET);    //file pointer 위치를 0으로 설정

            cnt\_line=0;

            while ((len = getline(&line, &size, fp2)) != -1) {

                    // printf("Retrieved line of length %zu :\n", len);

                    // printf("%s", line);

                    if(strchr(line,'R')!=NULL)

                        read[cnt\_line]=1;

                    if(strchr(line,'W')!=NULL)

                        write[cnt\_line]=1;

                    file\_reference\_string[cnt\_line++]=atoi(line);   //파일에 랜덤하게 생성한 참조 스트링을 읽어 배열에 저장.

            }

            free(line);  // getline에서 할당된 힙 메모리 해제

            // printf("page stream : ");

            // for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

            //     printf("%d ",file\_reference\_string[i]);

            // printf("\n");

            // for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

            //     printf("%d ",buf[i]);

            // printf("\n");

            // for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

            //     printf("%d ",read[i]);

            // printf("\n");

            // for (int i = 0; i < REFERENCE\_STRING\_NUM; i++)

            //     printf("%d ",write[i]);

            // printf("\n");

            fclose(fp2);

            execAlgo();

            // printf("case 2 : success\n");

            break;

        }

    }

    printf("D. 종료\n");

    return 0;

}

void esc1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>ESC<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>ESC<<<<<<<<<<<<<<<<<\n");

    int pointer=0;

    int pageFaults = 0;

    int check1,check2,check3,check4;

    int pageStringNum=REFERENCE\_STRING\_NUM;

    char c='?';

    int frame[pageframesNum];   //frame

    memset(frame, -1, sizeof(frame[0])\*pageframesNum);

    int r[pageframesNum];   //R bit

    memset(r, 1, sizeof(r[0])\*pageframesNum);

    int w[pageframesNum];   //W bit

    memset(w, 0, sizeof(w[0])\*pageframesNum);

    pageStringNum=REFERENCE\_STRING\_NUM;

    for(int i = 0; i < pageStringNum; i++)

    {

        check1=check2=check3=check4=0;

        for(int j = 0; j < pageframesNum; j++)

        {

            if(frame[j] == buf[i])  //page hit

            {

                r[j] = read[i]==1?1:r[j];   //set read bit

                w[j] = write[i]==1?1:w[j];     //set write bit

                check1=check2=check3=check4=1;

                c='H';

            }

        }

        if(!check1)  //page miss & no swapping

        {

            for(int j = 0; j < pageframesNum; ++j){

                if(frame[j] == -1){    //empty page

                    frame[j] = buf[i];   //just load page

                    r[j]=read[i];   //set r bit

                    w[j]=write[i];  //set w bit

                    r[j]=1;

                    check2 =check3=check4= 1;

                    pageFaults++;   //count pagefault

                    c='F';

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    break;

                }

            }

        }

        //goto

        LOOP:

        if(check2==0){  //page fault & swapping between (0,0) and incomming

            for(int j = 0; j < pageframesNum; ++j){

                if(((r[pointer]==0)&&(w[pointer]==0))) //if (0,0)

                {

                    frame[pointer] = buf[i];        //page replacement

                    r[pointer]=read[i];

                    w[pointer]=write[i];

                    r[pointer]=1;

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    check3=check4= 1;

                    pageFaults++;   //count pagefault

                    c='F';

                    break;

                }

                // r[pointer] = 0; //if r bit == 1 then r bit=0

                // w[pointer] = 0; //if w bit == 1 then w bit=0

                pointer = (pointer + 1) % pageframesNum;    //set queue pointer to next page

            }

        }

        if(check3==0){  //page fault & swapping between (0,1) and incomming while (1,0),(1,1)->(0,0),(0,1)

            for(int j = 0; j < pageframesNum; ++j){

                if((((r[pointer]==0)&&(w[pointer]==1)))) //check (0,1) is exist

                {

                    frame[pointer] = buf[i];        //page replacement

                    r[pointer]=read[i];

                    w[pointer]=write[i];

                    r[pointer]=1;

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    check4=1;

                    pageFaults++;   //count pagefault

                    c='F';

                    break;

                }

                else{

                    r[pointer] = 0; //if r bit == 1 then r bit=0

                    pointer = (pointer + 1) % pageframesNum;    //set queue pointer to next page

                }

            }

        }

        if(check4==0){

            //check2,3반복

            goto LOOP;

        }

        //print state

        printf("%2d(%d,%d)\t\t",buf[i],read[i],write[i]);  //print incomming page string

        fprintf(fp, "%2d(%d,%d)\t\t", buf[i],read[i],write[i]);

        for(int m = 0; m< pageframesNum; m++){

            // printf("%d(%d)\t",frame[m],second\_chance[m]);

            if(frame[m] != -1){

                printf(" %2d(%d,%d)\t\t", frame[m],r[m],w[m]);

                fprintf(fp, "%2d(%d,%d)\t\t", frame[m],r[m],w[m]);

            }else{

                printf(" -(-,-)\t\t");

                fprintf(fp, " -(-,-)\t\t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("ESC Total Page Faults : %d\n",pageFaults);

    fprintf(fp, "ESC Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

void sc1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>SC<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>SC<<<<<<<<<<<<<<<<<\n");

    int pointer=0;

    int pageFaults = 0;

    int pageStringNum=0;

    char c='?'; //c means page hit(h)/fault(f)

    int frame[pageframesNum];   //frame

    memset(frame, -1, sizeof(frame[0])\*pageframesNum);

    int second\_chance[pageframesNum];   //R bit

    memset(second\_chance, 0, sizeof(second\_chance[0])\*pageframesNum);

    pageStringNum=REFERENCE\_STRING\_NUM;

    for(int i = 0; i < pageStringNum; i++)

    {

        int check=0;

        for(int j = 0; j < pageframesNum; j++)

        {

            if(frame[j] == buf[i])  //page hit

            {

                second\_chance[j] = 1;   //set second chance bit = 1

                check=1;

                c='H';

            }

        }

        if(!check)  //page miss

        {

            while(1)

            {

                if(!second\_chance[pointer]) //if secod chance==0

                {

                    frame[pointer] = buf[i];        //page replacement

                    pointer=(pointer + 1) % pageframesNum;  //set queue pointer to next page

                    break;

                }

                second\_chance[pointer] = 0; //if second chance == 1 then second chance=0

                pointer = (pointer + 1) % pageframesNum;    //set queue pointer to next page

            }

            pageFaults++;   //count pagefault

            c='F';

        }

        //print state

        printf("%2d\t",buf[i]);  //print incomming page string

        fprintf(fp, "%2d\t", buf[i]);

        for(int m = 0; m< pageframesNum; m++){

            // printf("%d(%d)\t",frame[m],second\_chance[m]);

            if(frame[m] != -1){

                printf(" %2d(%d)\t", frame[m],second\_chance[m]);

                fprintf(fp, "%2d(%d)\t", frame[m],second\_chance[m]);

            }else{

                printf(" -(-) \t");

                fprintf(fp, " -(-) \t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("SC Total Page Faults : %d\n",pageFaults);

    fprintf(fp, "SC Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

void lfu1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>LFU<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>LFU<<<<<<<<<<<<<<<<<\n");

        int frequency[pageframesNum];   //counter

        memset(frequency, 0, pageframesNum\*sizeof(frequency[0]));

        int frames[pageframesNum];

        memset(frames, -1, pageframesNum\*sizeof(frames[0]));

        int len=REFERENCE\_STRING\_NUM;

        int pageFaults=0;

        char c='?';

        for(int i=0; i< len; i++){

            int check = 0;

            int min,minpos;

            for(int j=0; j<pageframesNum; j++){

                if (frames[j] == buf[i]){   //page hit

                    frequency[j]++; //increase frequency(counter)

                    check = 1;

                    c='H';

                    break;

                }

            }

            if(check == 0){ //page miss

                for(int j=0; j<pageframesNum; j++){

                    if(frames[j] == -1){    //no swapping. just insert page.

                        frames[j] = buf[i];

                        frequency[j];   //not increase frequency

                        check = 1;

                        pageFaults++;   //count page fault

                        c='F';

                        break;

                    }

                }

            }

            if(check == 0){ //page miss & swapping

                min=frequency[0];

                minpos=0;

                for(int k = 1; k< pageframesNum; k++){

                    if(frequency[k] < min){ //select least frequency page

                        min = frequency[k];

                        minpos = k;

                    }

                }

                frames[minpos] = buf[i];    //page swapping

                frequency[minpos] = 0;  //set frequency(counter)

                pageFaults++;   //count pagefault

                c='F';

            }

            //print state

            printf("%2d\t",buf[i]);  //print incomming page string

            fprintf(fp, "%2d\t", buf[i]);

            for(int m = 0; m< pageframesNum; m++){

                if(frames[m] != -1){

                    printf(" %2d(%d)\t", frames[m],frequency[m]);

                    fprintf(fp, "%2d(%d)\t", frames[m],frequency[m]);

                }else{

                    printf(" -(-) \t");

                    fprintf(fp, " -(-) \t");

                }

            }

            printf("%c\n",c);

            fprintf(fp, "%c\n",c);

        }

    printf("LFU Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "LFU Total Page Faults : %d\n", pageFaults);

    fclose(fp);

}

void lru1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>LRU<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>LRU<<<<<<<<<<<<<<<<<\n");

    int pageStringNum=REFERENCE\_STRING\_NUM;

    int frames[pageframesNum];

    int temp[pageframesNum];

    int counter = 0;

    int check1, check2, pageFaults = 0;

    char c='?';

    for(int i = 0; i < pageframesNum; ++i){  //initialize

        frames[i] = -1;

    }

    for(int i = 0; i < pageStringNum; ++i){

        check1 = 0;

        check2 = 0;

        for(int j = 0; j < pageframesNum; ++j){

            if(frames[j] == buf[i]){  //page hit

                counter++;  //increase time(=counter)

                temp[j] = counter;  //record time(=counter)

                check1 = 1;

                check2 = 1;

                c='H';

                break;

            }

        }

        if(check1 == 0){     //page fault & no swapping(just insert page)

            for(int j = 0; j < pageframesNum; ++j){

                if(frames[j] == -1){    //empty page

                    counter++;  //increase time(=counter)

                    pageFaults++;   //count page fault

                    frames[j] = buf[i];   //page replacement

                    temp[j] = counter;  //record time(=counter)

                    check2 = 1;

                    c='F';

                    break;

                }

            }

        }

        if(check2 == 0){     //page fault & swapping

            int minimum = temp[0];

            int pos=0;

            for(int j = 1; j < pageframesNum; ++j){

                if(temp[j] < minimum){  //select old used page

                    minimum = temp[j];

                    pos = j;    //old used page's index

                }

            }

            counter++;  //increase time(=counter)

            pageFaults++;   //count page fault

            frames[pos] = buf[i]; //page replacement

            temp[pos] = counter;    //record time(=counter)

            c='F';

        }

        //print state

        printf("%2d\t",buf[i]);  //print incomming page string

        fprintf(fp, "%2d\t", buf[i]);

        for(int j = 0; j < pageframesNum; ++j){  //print pages state

            if(frames[j] != -1){

                printf("%2d\t", frames[j]);

                fprintf(fp, "%2d\t", frames[j]);

            }else{

                printf(" - \t");

                fprintf(fp, " - \t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("LRU Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "LRU Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

void optimal1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

     printf(">>>>>>>>>>>>>>>>>>>>Optimal<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>Optimal<<<<<<<<<<<<<<<<<\n");

    int pageStringNum=REFERENCE\_STRING\_NUM;

    int frames[pageframesNum], temp[pageframesNum];

    int check1, check2, check3;

    int pos, max;

    int pageFaults = 0;

    char c='?';

    for(int i = 0; i < pageframesNum; ++i){

        frames[i] = -1;

    }

    for(int i = 0; i < pageStringNum; ++i){

        check1 = 0;

        check2 = 0;

        for(int j = 0; j < pageframesNum; ++j){

            if(frames[j] == buf[i]){    //page hit

                   check1 = 1;

                   check2 = 1;

                   c='H';

                   break;

               }

        }

        if(check1 == 0){ //page miss & no swapping(just insert page)

            for(int j = 0; j < pageframesNum; ++j){

                if(frames[j] == -1){    //no swap

                    pageFaults++;

                    frames[j] = buf[i]; //insert page

                    check2 = 1;

                    c='F';

                    break;

                }

            }

        }

        if(check2 == 0){ //page miss & swapping

            check3 =0;

            for(int j = 0; j < pageframesNum; ++j){

                temp[j] = -1;

                for(int k = i + 1; k < pageStringNum; ++k){

                    if(frames[j] == buf[k]){

                        temp[j] = k;    //현재 프레임에 저장된 string이 이후에 등장하명 해당 index 저장, 아니면 -1.

                        break;

                    }

                }

            }

            for(int j = 0; j < pageframesNum; ++j){

                if(temp[j] == -1){  //optimal, 현재 프레임에 저장된 string 중 이후에 나오지 않는 string 존재.

                    pos = j;

                    check3 = 1;

                    break;

                }

            }

            if(check3 ==0){  //optimal, 현재 프레임에 저장된 string이 모두 이후에 등장함.

             max = temp[0];

             pos = 0;

                for(int j = 1; j < pageframesNum; ++j){

                    if(temp[j] > max){  //optimal, 현재 프레임에 저장된 string 중 가장 나중에 등장하는 string 찾기.

                    max = temp[j];

                    pos = j;

                    }

                }

            }

            frames[pos] = buf[i];   //가장 나중에 사용될 string 을 page replacement

            pageFaults++;   //count page fault

            c='F';

        }

        //print state

        printf("%2d\t",buf[i]);

        fprintf(fp, "%2d\t", buf[i]);

        for(int j = 0; j < pageframesNum; ++j){

            if(frames[j] != -1){

                printf(" %2d\t", frames[j]);

                fprintf(fp, "%2d\t", frames[j]);

            }else{

                printf(" - \t");

                fprintf(fp, " - \t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("Optimal Total Page Faults:\t%d\n", pageFaults);

    fprintf(fp, "Optimal Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}

void lifo1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>LIFO<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>LIFO<<<<<<<<<<<<<<<<<\n");

    int pageFaults = 0;

    int check;

    int pageStringNum=REFERENCE\_STRING\_NUM;

    char c='?';

    int stack[pageframesNum];   //create stack

    int top=-1;

    for(int m = 0; m < pageframesNum; m++)

    {

        stack[m] = -1;

    }

    for(int m = 0; m < pageStringNum; m++)

    {

        check=0;

        for(int n = 0; n < pageframesNum; n++) //PAGE FAULT 검사

        {

            if(buf[m] == stack[n])    //PAGE HIT

            {

                check=1;

                c='H';

            }

        }

        //FAULT인 경우 올바른 FRAME 위치에 저장

        if((pageFaults < pageframesNum) && (check == 0))  //PAGE FAULT&&NO SWAPPING

        {

            stack[++top]= buf[m];   //INSERT PAGE INTO STACK

            pageFaults++;   //COUNT PAGE FAULT

            c='F';

        }

        else if(check == 0)     //PAGE FAULT&SWAPPING

        {

            stack[top]= buf[m];     //INSERT PAGE INTO TOP OF STACK

            pageFaults++;   //COUNT PAGE FAULT

            c='F';

        }

        //PRINT STATE

        printf("%2d\t",buf[m]);

        fprintf(fp, "%2d\t", buf[m]);

        for(int n = 0; n < pageframesNum; n++)

        {

            if(stack[n] != -1){

                printf("%2d\t", stack[n]);

                fprintf(fp, "%2d\t", stack[n]);

            }else{

                printf(" - \t");

                fprintf(fp, " - \t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("LIFO Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "LIFO Total Page Faults : %d\n", pageFaults);

    fclose(fp);

}

void fifo1(){

    FILE \*fp;

    if((fp=fopen("result.txt","a+"))==NULL){   //읽기 쓰기 모두 가능하도록 오픈. 이미 파일이 존재하면 파일 길이 0이 됨. 파일이 존재 하지 않으면 creat.

        fprintf(stderr,"fopen error for %s\n","result.txt");

        exit(1);

    }

    printf(">>>>>>>>>>>>>>>>>>>>FIFO<<<<<<<<<<<<<<<<<\n");

    fprintf(fp, ">>>>>>>>>>>>>>>>>>>>FIFO<<<<<<<<<<<<<<<<<\n");

    int pageFaults = 0;

    int check;

    int pageStringNum= REFERENCE\_STRING\_NUM;

    char c='?';

    int frame[pageframesNum];

    for(int m = 0; m < pageframesNum; m++)

    {

        frame[m] = -1;

    }

    for(int m = 0; m < pageStringNum; m++)

    {

        check = 0;

        for(int n = 0; n < pageframesNum; n++)

        {

            if(buf[m] == frame[n])  //psge hit

            {

                check=1;

                c='H';

            }

        }

        if((pageFaults < pageframesNum) && (check == 0)){   //page fault&no swapping(just insert page)

            pageFaults++;   //count page fault

            frame[(pageFaults - 1) % pageframesNum] = buf[m];  //insert page

            c='F';

        }else if(check == 0) {  //fault&swapping

            pageFaults++;   //count page fault

            frame[(pageFaults - 1) % pageframesNum] = buf[m];   //page replacement

            c='F';

        }

        //print state

        printf("%2d\t",buf[m]);

        fprintf(fp, "%2d\t", buf[m]);

        for(int n = 0; n < pageframesNum; n++)

        {

            if(frame[n] != -1){

                printf(" %2d\t", frame[n]);

                fprintf(fp, "%2d\t", frame[n]);

            }else{

                printf(" - \t");

                fprintf(fp, " - \t");

            }

        }

        printf("%c\n",c);

        fprintf(fp, "%c\n",c);

    }

    printf("FIFO Total Page Faults : %d\n", pageFaults);

    fprintf(fp, "FIFO Total Page Faults : %d\n", pageFaults);

    fclose(fp);

    return;

}