❋ سیستم‌عامل موردنیاز:

Mac: macos 10.11 or later

Windows: Windows 7/8/10

Linux: Ubuntu 16.04 or later

❋ تکنولوژی و زبان مورد استفاده و ورژن آن:

C++: ISO/IEC 14882:2020 (C++20)

❋ کتابخانه‌هّای مورد استفاده:

iostream
vector

❋ تشریح دقیق و روشن مراحل حل مسئله:

توضیح main method :

در ابتدا ورودی‌ها را منطبق با خواسته‌ی مسئله دریافت میکنیم. اکنون باید یک شماره‌ای را به عنوان حدس به j نسبت دهیم. باید مطمئن باشیم که پاسخ نهاییِ j بین این حدس و کالای اول قرار ندارد. این حدس توسط تابع GuessJ انجام می‌شود. سپس این حدس (guessAboutJ) را توسط تابع Check راستی‌آزمایی میکنیم. اگر با شروع کردن از جنس شماره‌یj اُم طبق خواسته و قوانین مسئله، توانستیم تا آخرین کالا را داخل جعبه‌ها قرار دهیم این تابع true و اگر جعبه‌ها تمام شوند و ما هنوز کالا در دست داشته باشیم، false بر می‌گرداند. از نقیض این تابع در داخل شرط حلقه‌ی while استفاده میکنیم تا اگر حدس ما غلط بود و جعبه کم آمد، در داخل دستورات حلقه یکی به j اضافه کنیم (یعنی از یک کالا دیرتر شروع به جا دادن داخل جعبه‌ها بکنیم) تا به پاسخ نهایی برسیم. هنگامی که j پیدا شد (یعنی شماره‌ای از کالاها پیدا شده که از آن تا آخرین کالا در داخل جعبه‌ها طبق قوانین جا میگیرد)، خواسته‌ی مسئله که تعداد کالاهای جاگرفته در جعبه میباشد را با فرمول تعداد کل کالاها منهای j به دست آورده (numberOfObjectsThatArePutInBoxes) و نمایش می‌دهیم.

توضیح تابع GuessJ :

میدانیم در بهترین حالت وقتی کالاها را داخل جعبه‌ها میگذاریم، هیچ فضای خالی در جعبه‌ها نخواهد ماند. پس بیشترین اندازه‌ای از کالاها که میتوانیم در جعبه‌ها جا دهیم مساوی اندازه‌ی هر جعبه ضربدر تعداد جعبه‌ها خواهد بود. بنابراین در داخل این تابع اندازه‌ی هر کالا را از انتها شروع به جمع کردن میکنیم (sum) تا زمانی که حاصل به مجموع ظرفیت جعبه‌ها برسد. هنگامی که sum به مقدار

مطلوب رسید، شماره‌ی آخرین کالایی که اندازه‌ی آن را جمع زدیم را به عنوان حدس شماره‌ی کالا برای شروع پروسه توسط تابع بر می‌گردانی و در متغییر guessAboutJ ذخیره میکنیم.

توضیح تابع Check :

این تابع باید از کالای شماره‌ی guess ام که به عنوان یکی از پارامترهای ورودی دریافت میکنید تا آخرین کالا را داخل جعبه‌ها طبق قوانین جاگذاری کند و اگر آخرین کالا نیز در جعبه‌ها جا شد، true و هر حالتی جز این را false برگرداند. برای اینکار یک vector تعریف میکنیم که نشان‌دهنده‌ی فضای خالی هر جعبه می‌باشد سپس حلقه‌ی for را استفاده میکنیم تا به نوبت جعبهّآ را پر کنیم. متغیر idOfInProgressObject نشان‌دهنده‌ی شماره‌ی کالایی است که در دست داریم و می‌خواهیم در یک جعبه قرار دهیم. در ادامه حلقه‌ی while به ترتیب کالاها را چک می‌کند که در جعبه‌ی حال حاظر قرار می‌گیرد یا خیر و این عمل با کم کردن اندازه‌ی کالایی که در دست داریم از فضای خالی جعبه‌ی مورد نظر انجام می‌گیرد. اگر اندازه‌ی فضای خالی جعبه مساوی صفر شود، متغیر temp را false کرده تا از while بیرون آمده و به سراغ جعبه‌ی بعدی می‌رویم. اگر هم فضای خالی جعبه کمتر از صفر شود، دوباره آخرین کالایی که اندازه‌اش را از این فضا کم کرده بودیم را به emptySpaceOfBoxes مربوط به آن جعبه اضافه می‌کنیم و با تغییر متغیر temp از while خارج می‌شویم اما میدانیم که در دفعه‌ی بعدی باید با همین کالا را در داخل جعبه قرار دهیم (خط ۸۲). هر جا که هر جا که آخرین کالا در یکی از جعبه‌ها جا گیرد، ارزش متغیر idOfInProgressObject برابر با تعداد تعداد کل کالاها خواهد شد که ما نیز از همین شرط برای تعیین ارزشی که باید بازگردانیم استفاده می‌کنیم.

❋ کامپایلر مورد نیاز و ورژن آن:

Mac: Xcode 10 or later
Windows: Code::Blocks 16.01 or later
Linux: GNU GCC, gEdit

❋ لینک سورس کد روی github :

https://github.com/behnamgrowthpath/fanavard-challenge.git