معین کرمی 810198540

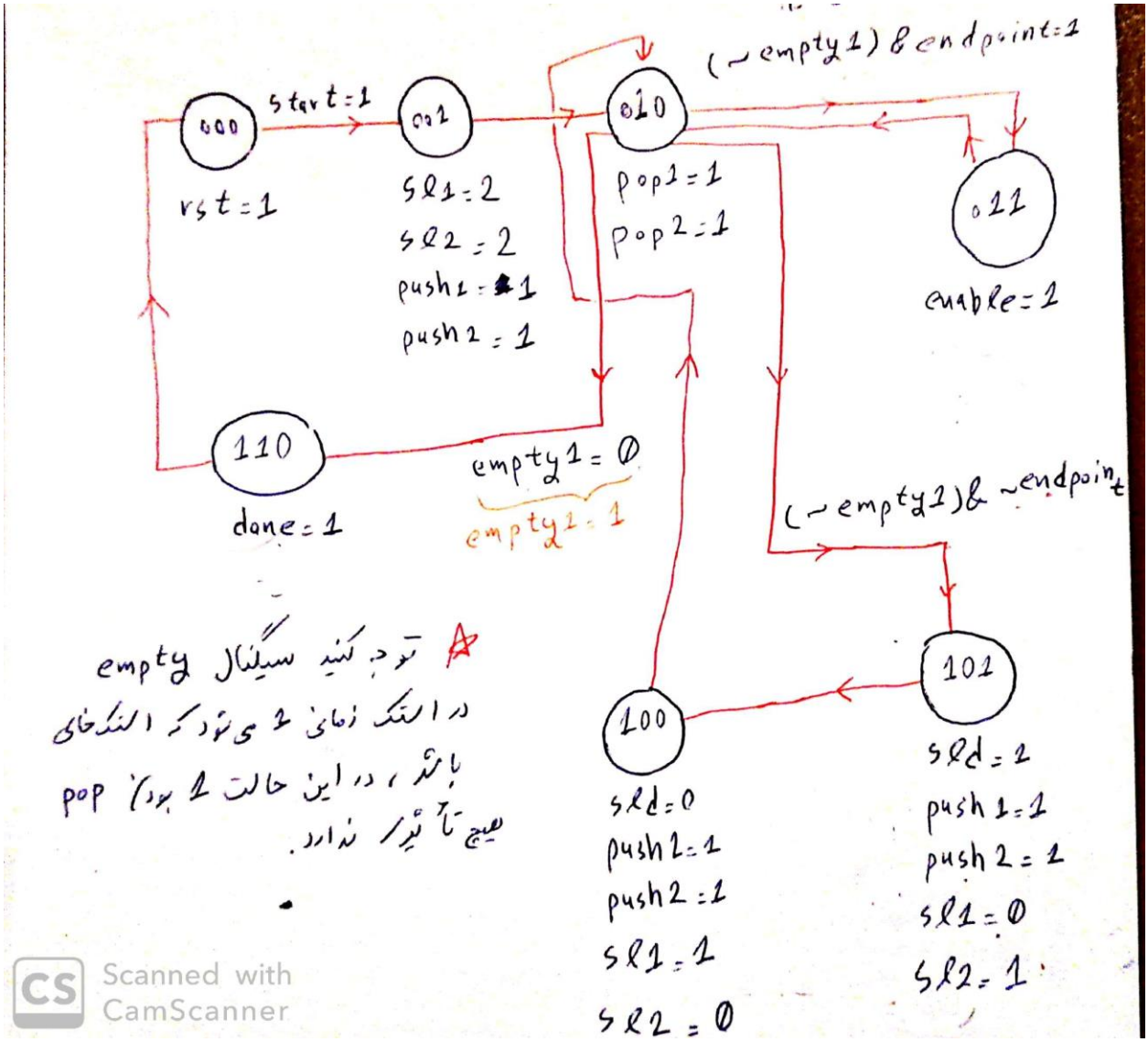
**توجه: برای بهینه سازی و ساده تر شدن کد چون سیگنال های کنترلی دو استک در تمام مراحل یکسان بودند این دو را چه در data path و چه در controller با هم ترکیب کرده و یکی در نظر می‌گیریم. یعنی:

(push1 = push2 = push, top1 = top2 = top, pop1 = pop2 = pop, empty1 = empty2 = empty)

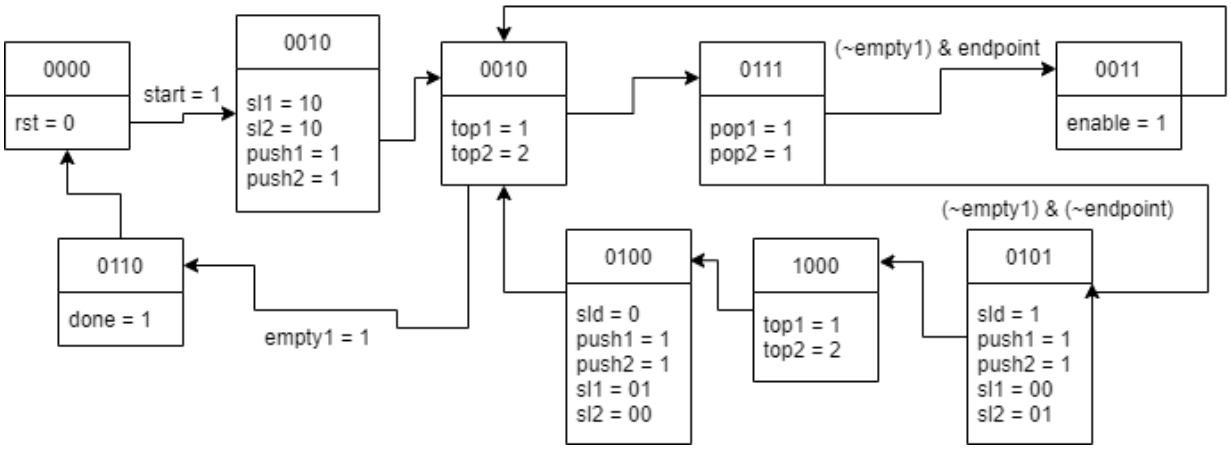
ایرادات state machine digram :

(توجه کنید که برای خوانایی بیشتر به استیت ها در این مرحله یک شماره داده شده است)

1. در ابتدا توجه کنید شرط رفتن از استیت 0010 به 0110 این است که empty = 1 که به اشتباه نوشته شده است empty = 0.
2. نیاز است بعد از استیت 0010 یک استیت دیگر داشته باشیم تا مقادیر سر استک بر روی خروجی آن ها قرار بگیرند که این استیت در شکل کشیده نشده است.
   راه حل: باید یک استیت در اینجا اضافه شود که از استیت 0010 به آن وارد شویم و طریقه ی خارج شدن از آن دقیقا مانند طریقه ی خارج شدن از استیت 0010 در حالت قبل است.
3. بعد از استیت 0101 که می‌خواهیم عنصر سر استک اول (n - 1) را دوباره به داخل استک پوش کنیم (طبق رابطه بازگشتی) مقدار n روی خروجی استک قرار دارد.(ما می‌خواهیم n-1 , m-1 را داخل استک ها بریزیم و در مرحله قبل (استیت 0101) مقادیر n-1 , m را داخل استک ریخته ایم). ولی ما به مقدار n – 1 نیاز داریم.
   راه حل: به یک استیت اضافی نیاز داریم که در آن با دستور top بتوانیم مقدار سر استک‌ها را بخوانیم و سپس به استیت 0100 برویم و مقادیر را داخل آن‌ها پوش کنیم.
4. به دلیل اضافه کردن ۲ استیت که در بالا گفته شد ما در مجموع ۹ استیت خواهیم داشت پی مجبوریم شماره استیت‌ها را ۴بیتی در نظر بگیریم.

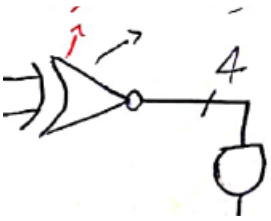


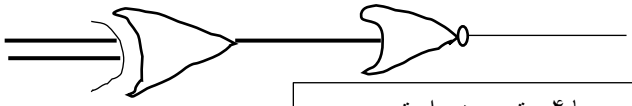
استیت ماشین دیاگرام قدیمی



استیت ماشین دیاگرام جدید

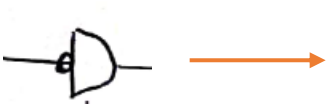
ایرادات مسیر داده: مسیر داده ایرادی نداشت و تقریبا بدون تغییر باقی ماند فقط برای ساده تر شدن کد از یک سری المان های دیگر استفاده شد که در زیر آمده است:

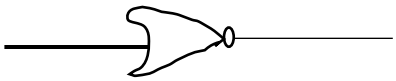




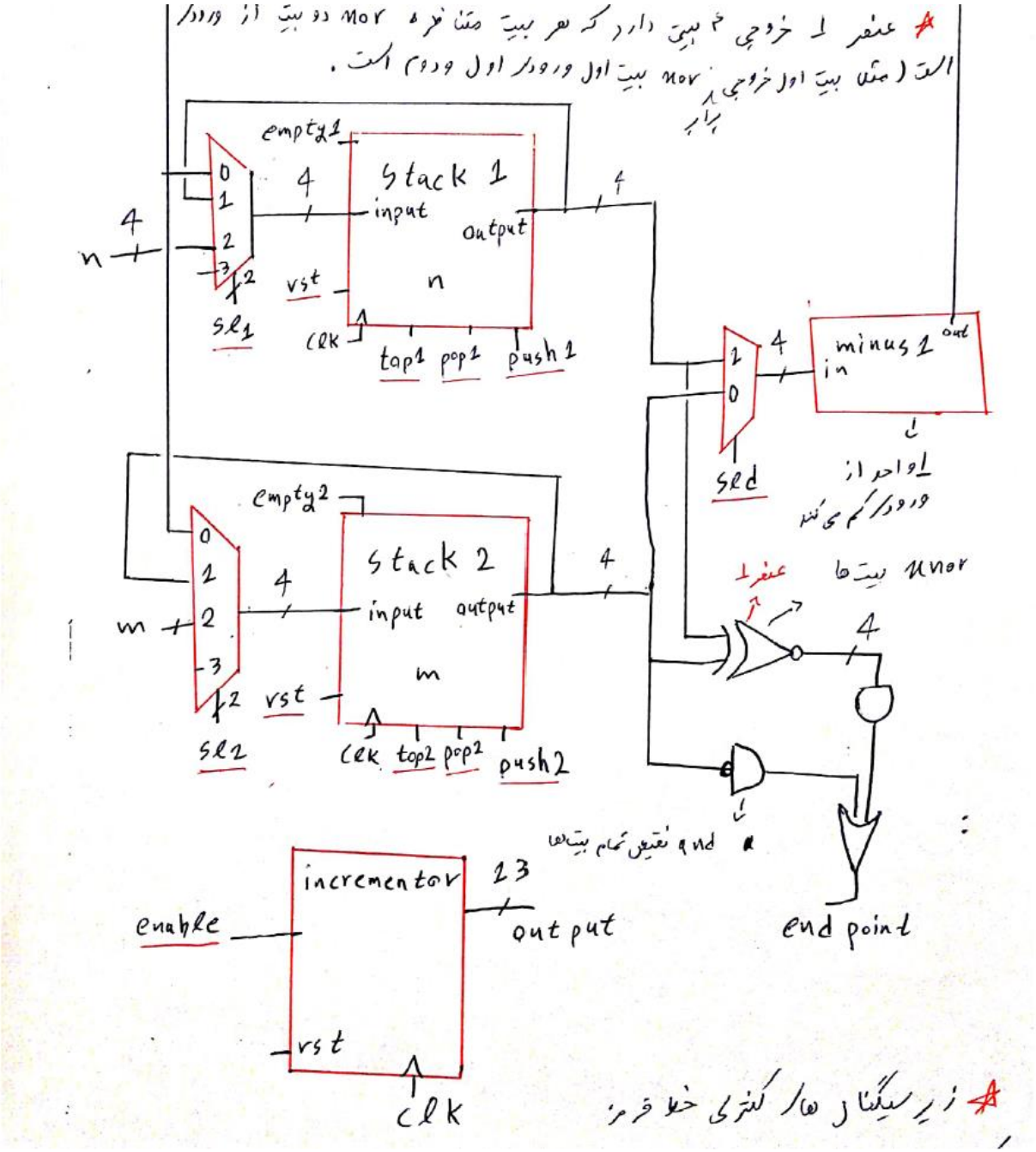
این المنت یک گیت xor است که دو عدد ۴ بیتی دریافت می‌کند و xor آن ها که یک عدد ۴ بیتی است را خروجی می‌دهد.

این یک المنت nor با ۴ بیت ورودی است



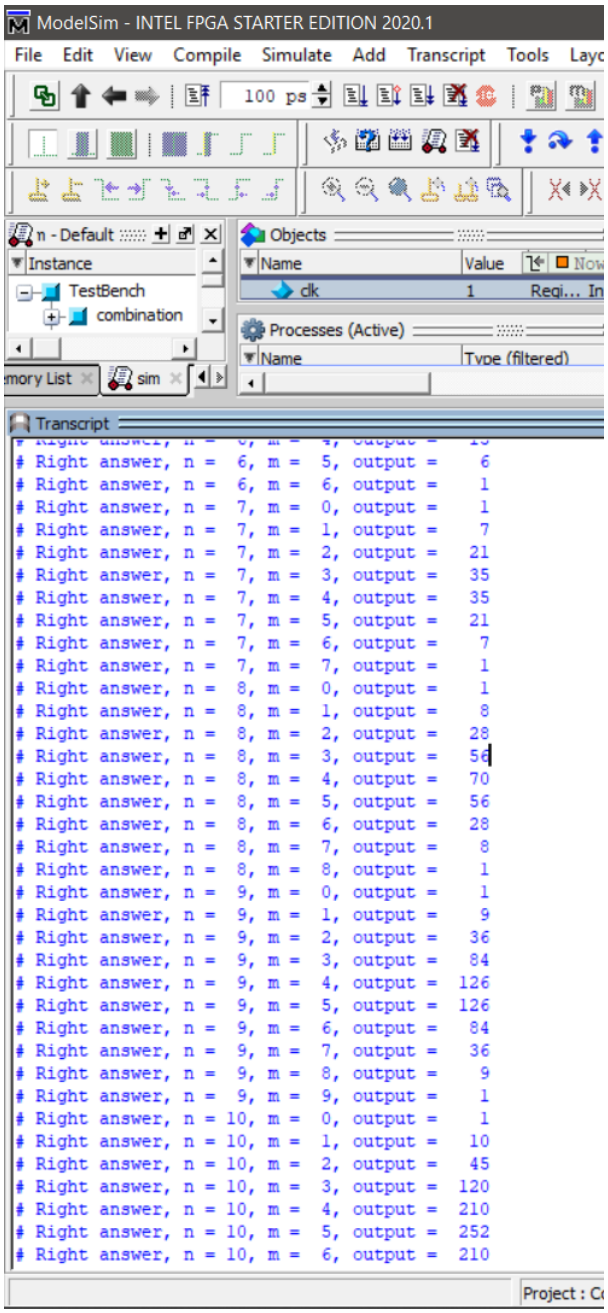


این یک المنت nor با ۴ بیت ورودی است

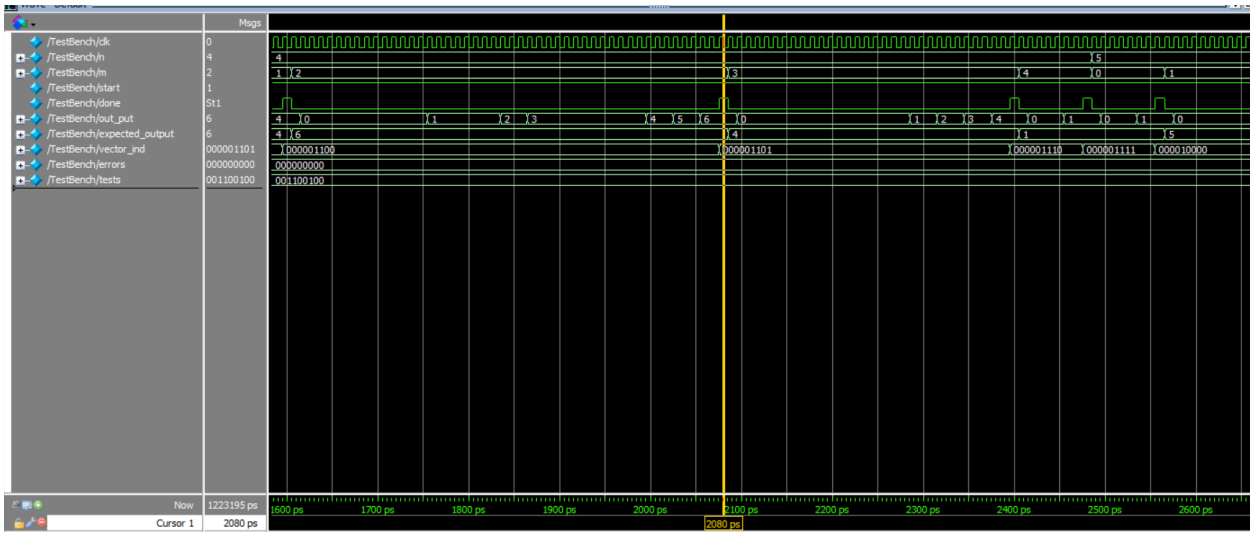

empty1
stack 1
input
output
n
rst
clk
tap1
pop1
push1
sel1
4
n
empty2
stack 2
input
output
m
rst
clk
top2
pop2
push2
sel2
m
minus 1
in
out
sel
enable
incrementor
23
out put
rst
clk
end point

کل دیتاپث

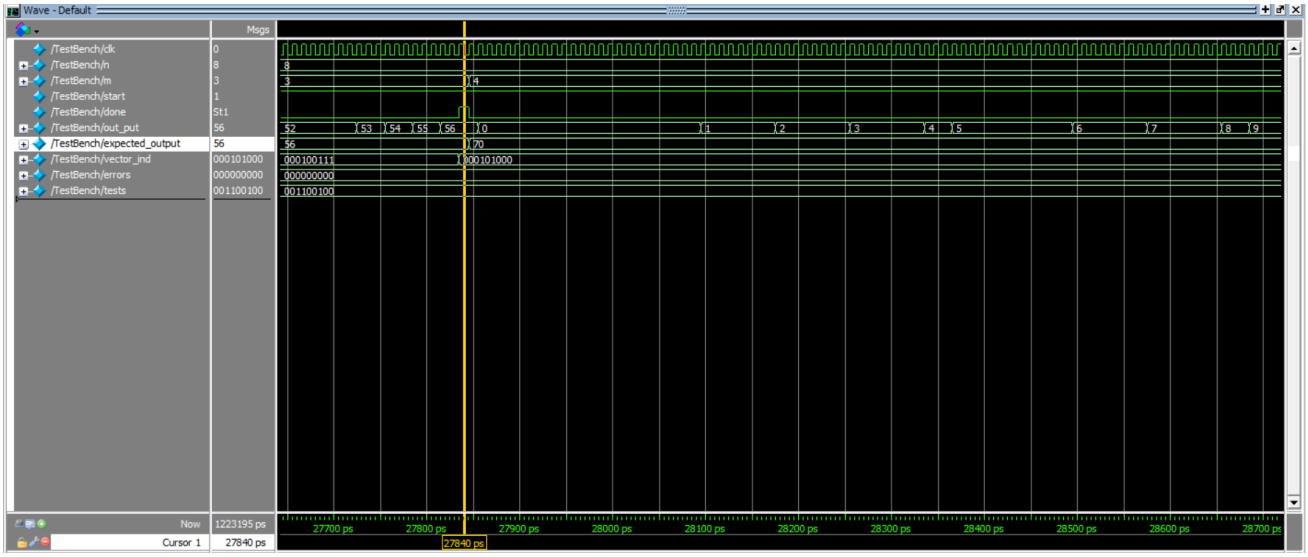
تست درستی: برای چک کردن درستی طراحی از یک تست بنچ self checking استفاده شده است که ورودی ها و خروجی های مورد انتظار آن با یک کد به زبان c++ که داخل پوشه sim/model است ساخته شده است. این تست بنچ ورودی های مختلف (۱۰۰ ورودی) را به صورت اتوماتیک به مدار می‌دهد و خروجی آن را با خروجی مورد انتظار چک می‌کند. در زیر می‌توانید تصاویر تست های مختلف را ببینید.



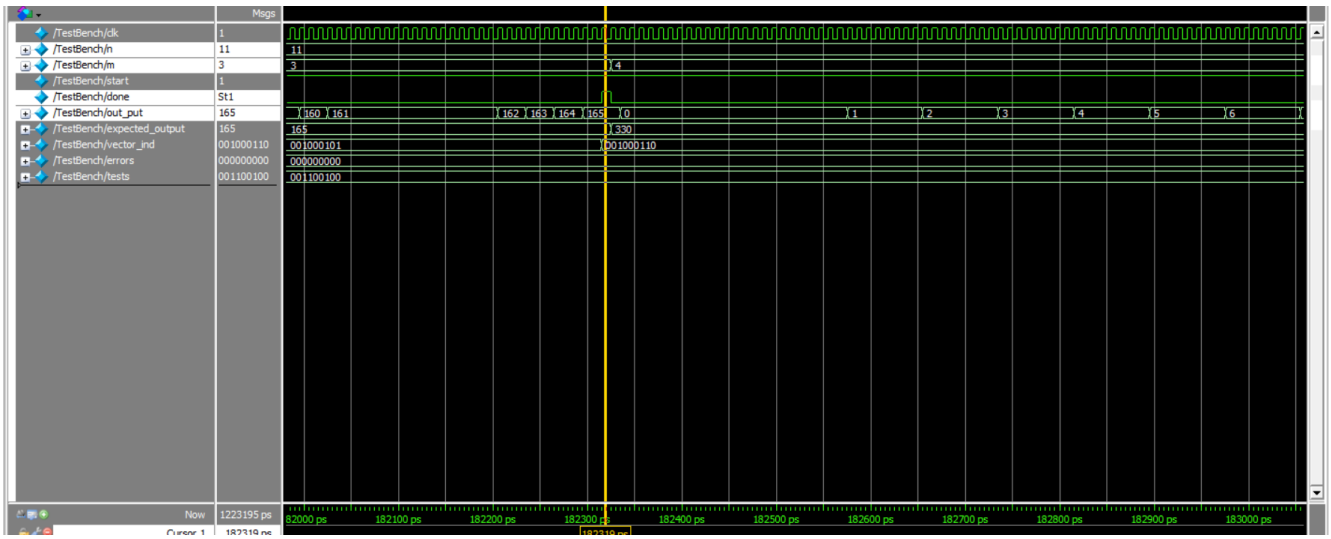
نتایج برخی تست ها



نتایج تست به ازای n = 4, m = 2 جواب = 6



نتیجه تست به ازای n = 8 , m = 3 جواب = 56



نتیجه تست به ازای n = 11, m = 3, جواب = 165