Alessio Barletta 5686474

Compiti 8.2, 9.1, 9.2

Compito 8.2 Pochi bugiardi

Ho creato la funzione MCPrimalityTest che mi restituisce 1 se il numero è probabilmente primo, 0 altrimenti.

Passo alla funzione MCPrimalityTest due argomenti, il numero di Carmichael e un numero a che va da 2 a n-2 con n uguale ai primi sette numeri di Carmichael, se la funzione mi ritorna 1 allora ho trovato che a è un bugiardo in quanto mi restituisce primo anche se non è vero. Successivamente salvo tutti i numeri in un file dedicato.

Possiamo notare come per ogni numero di Carmichael i bugiardi siano comunque decisamente meno della metà, rendendo l'algoritmo MCPrimalityTest ottimo anche per questi numeri.

Per 561 i bugiardi sono 8:

> 50; 101; 103; 256; 305; 458; 460; 511.

Per 1105 i bugiardi sono 28:

> 47; 72; 98; 123; 132; 242; 256; 268; 293; 327; 341; 353; 463; 548; 557; 642; 752; 764; 778; 812; 837; 849; 863; 973; 982; 1007; 1033; 1058.

Per 1729 i bugiardi sono 161:

> 9; 10; 12; 16; 69; 74; 75; 81; 90; 92; 100; 103; 108; 120; 129; 144; 160; 166; 172; 173; 181; 191; 192; 235; 256; 257; 263; 282; 289; 302; 335; 347; 355; 363; 365; 374; 376; 386; 402; 426; 433; 438; 439; 443; 484; 493; 529; 536; 545; 555; 562; 563; 568; 575; 584; 621; 625; 638; 649; 653; 654; 666; 675; 690; 699; 706; 729; 740; 750; 757; 797; 802; 807; 809; 810; 828; 829; 831; 841; 857; 872; 888; 898; 900; 901; 919; 920; 922; 927; 932; 972; 979; 989; 1000; 1023; 1030; 1039; 1054; 1063; 1075; 1076; 1080; 1091; 1104; 1108; 1145; 1154; 1161; 1166; 1167; 1174; 1184; 1193; 1200; 1236; 1245; 1286; 1290; 1291; 1296; 1303; 1327; 1343; 1353; 1355; 1364; 1366; 1374; 1382; 1394; 1427; 1440; 1447; 1466; 1472; 1473; 1494; 1537; 1538; 1548; 1556; 1557; 1563; 1569; 1585; 1600; 1609; 1621; 1626; 1629; 1637; 1639; 1648; 1654; 1655; 1660; 1713; 1717; 1719; 1720.

Per 2465 i bugiardi sono 68:

> 47; 72; 98; 157; 217; 242; 254; 256; 293; 302; 327; 387; 426; 438; 472; 548; 582; 596; 693; 727; 752; 888; 897; 934; 1007; 1019; 1033; 1058; 1092; 1143; 1152; 1177; 1203; 1228; 1237; 1262; 1288; 1313; 1322; 1373; 1407; 1432; 1446; 1458; 1531; 1568; 1577; 1713; 1738; 1772; 1869; 1883; 1917; 1993; 2027; 2039; 2078; 2138; 2163; 2172; 2209; 2211; 2223; 2248; 2308; 2367; 2393; 2418.

Per 2821 i bugiardi sono 268:

> 9; 12; 16; 17; 75; 81; 100; 107; 108; 113; 144; 153; 165; 166; 181; 183; 191; 192; 199; 204; 211; 220; 235; 244; 251; 256; 272; 285; 289; 290; 295; 326; 363; 373; 380; 386; 402; 417; 426; 433; 438; 443; 446; 484; 517; 529; 530; 534; 536; 547; 555; 556; 562; 568; 584; 599; 615; 625; 627; 638; 647; 649; 653; 654; 659; 675; 690; 699; 706; 712; 718; 719; 729; 740; 766; 790; 797; 807; 818; 820; 829; 836; 841; 872; 881; 893; 894; 900; 919; 920; 922; 932; 939; 963; 972; 985; 989; 1002; 1010; 1013; 1017; 1018; 1030; 1076; 1082; 1083; 1091; 1093; 1101; 1102; 1109; 1121;

1166; 1174; 1192; 1193; 1195; 1200; 1221; 1252; 1275; 1284; 1286; 1290; 1296; 1303; 1327; 1336; 1353; 1356; 1366; 1374; 1375; 1377; 1444; 1446; 1447; 1455; 1465; 1468; 1485; 1494; 1518; 1525; 1531; 1535; 1537; 1546; 1569; 1600; 1621; 1626; 1628; 1629; 1647; 1655; 1700; 1712; 1719; 1720; 1728; 1730; 1738; 1739; 1745; 1791; 1803; 1804; 1808; 1811; 1819; 1832; 1836; 1849; 1858; 1882; 1889; 1899; 1901; 1902; 1921; 1927; 1928; 1940; 1949; 1980; 1985; 1992; 2001; 2003; 2014; 2024; 2031; 2055; 2081; 2092; 2102; 2103; 2109; 2115; 2122; 2131; 2146; 2162; 2167; 2168; 2172; 2174; 2183; 2194; 2196; 2206; 2222; 2237; 2253; 2259; 2265; 2266; 2274; 2285; 2287; 2291; 2292; 2304; 2337; 2375; 2378; 2383; 2388; 2395; 2404; 2419; 2435; 2441; 2448; 2458; 2495; 2526; 2531; 2532; 2536; 2549; 2565; 2570; 2577; 2586; 2601; 2610; 2617; 2622; 2629; 2630; 2638; 2640; 2655; 2656; 2668; 2677; 2708; 2713; 2714; 2721; 2740; 2746; 2804; 2805; 2809; 2812.

Per 6601 i bugiardi sono 328:

16; 18; 40; 45; 66; 78; 100; 122; 141; 165; 195; 242; 250; 256; 286; 288; 303; 305; 318; 324; 327; 332; 338; 433; 474; 482; 508; 517; 523; 573; 605; 611; 619; 625; 640; 652; 715; 720; 739; 769; 795; 804; 810; 821; 824; 830; 843; 845; 877; 898; 927; 961; 983; 1002; 1056; 1082; 1091; 1108; 1111; 1117; 1132; 1147; 1166; 1185; 1188; 1193; 1199; 1205; 1248; 1270; 1289; 1313; 1369; 1378; 1404; 1417; 1451; 1453; 1466; 1513; 1527; 1576; 1581; 1600; 1630; 1644; 1671; 1682; 1691; 1721; 1738; 1753; 1762; 1767; 1773; 1779; 1788; 1800; 1868; 1887; 1931; 1952; 1964; 1972; 1993; 2008; 2010; 2025; 2027; 2054; 2060; 2075; 2101; 2109; 2131; 2174; 2196; 2204; 2218; 2245; 2256; 2259; 2312; 2347; 2396; 2483; 2491; 2505; 2526; 2538; 2543; 2567; 2584; 2601; 2614; 2634; 2640; 2661; 2705; 2729; 2770; 2813; 2830; 2869; 2888; 2907; 2915; 2927; 2936; 2948; 2962; 2970; 3057; 3079; 3091; 3117; 3120; 3139; 3156; 3175; 3188; 3202; 3249; 3298; 3303; 3352; 3399; 3413; 3426; 3445; 3462; 3481; 3484; 3510; 3522; 3544; 3631; 3639; 3653; 3665; 3674; 3686; 3694; 3713; 3732; 3771; 3788; 3831; 3872; 3896; 3940; 3961; 3967; 3987; 4000; 4017; 4034; 4058; 4063; 4075; 4096; 4110; 4118; 4205; 4254; 4289; 4342; 4345; 4356; 4383; 4397; 4405; 4427; 4470; 4492; 4500; 4526; 4541; 4547; 4574; 4576; 4591; 4593; 4608; 4629; 4637; 4649; 4670; 4714; 4733; 4801; 4813; 4822; 4828; 4834; 4839; 4848; 4863; 4880; 4910; 4919; 4930; 4957; 4971; 5001; 5020; 5025; 5074; 5088; 5135; 5148; 5150; 5184; 5197; 5223; 5232; 5288; 5312; 5331; 5353; 5396; 5402; 5408; 5413; 5416; 5435; 5454; 5469; 5484; 5490; 5493; 5510; 5519; 5545; 5599; 5618; 5640; 5674; 5703; 5724; 5756; 5758; 5771; 5777; 5780; 5791; 5797; 5806; 5832; 5862; 5881; 5886; 5949; 5961; 5976; 5982; 5990; 5996; 6028; 6078; 6084; 6093; 6119; 6127; 6168; 6263; 6269; 6274; 6277; 6283; 6296; 6298; 6313; 6315; 6345; 6351; 6359; 6406; 6436; 6460; 6479; 6501; 6523; 6535; 6556; 6561; 6583; 6585.

Per 8911 i bugiardi sono 1780:

3; 4; 9; 12; 13; 16; 23; 25; 27; 31; 34; 36; 39; 41; 48; 52; 64; 69; 75; 81; 92; 93; 94; 97; 100; 102; 106; 108; 110; 117; 121; 123; 124; 136; 144; 145; 146; 149; 156; 158; 163; 164; 166; 169; 185; 192; 207; 208; 218; 225; 226; 243; 256; 262; 263; 276; 277; 279; 282; 289; 291; 299; 300; 305; 306; 314; 318; 324; 325; 330; 351; 355; 358; 359; 363; 368; 369; 372; 376; 383; 388; 389; 398; 400; 403; 408; 409; 424; 430; 432; 433; 435; 438; 440; 442; 447; 457; 468; 473; 474; 484; 489; 491; 492; 496; 498; 505; 507; 521; 529; 533; 535; 544; 555; 557; 562; 563; 566; 571; 575; 576; 580; 584; 590; 596; 613; 617; 621; 624; 625; 632; 635; 638; 649; 652; 654; 656; 661; 662; 664; 674; 675; 676; 677; 678; 695; 698; 709; 713; 729; 740; 746; 755; 758; 768; 775; 782; 786; 787; 789; 794; 799; 802; 811; 814; 821; 823; 828; 831; 832; 837; 838; 841; 846; 850; 857; 867; 872; 873; 890; 897; 900; 901; 904; 915; 918; 922; 934; 935; 941; 942; 943; 947; 954; 958; 961; 965; 967; 972; 974; 975; 979; 983; 990; 1024; 1025; 1030; 1031; 1039; 1048; 1052; 1053; 1054; 1055; 1061; 1063; 1065; 1073; 1074; 1077; 1087; 1089; 1094; 1104; 1107; 1108; 1116; 1123; 1128; 1133; 1145; 1149; 1156; 1163; 1164; 1167; 1181; 1194; 1196; 1198; 1200; 1201; 1209; 1220;

1222; 1224; 1227; 1237; 1238; 1241; 1249; 1255; 1256; 1261; 1271; 1272; 1282; 1286; 1289; 1290; 1291; 1296; 1297; 1299; 1300; 1305; 1307; 1314; 1319; 1320; 1321; 1326; 1327; 1341; 1342; 1343; 1346; 1355; 1366; 1369; 1370; 1371; 1373; 1378; 1382; 1394; 1404; 1411; 1419; 1420; 1422; 1423; 1427; 1429; 1430; 1432; 1436; 1438; 1452; 1459; 1467; 1472; 1473; 1476; 1488; 1493; 1494; 1499; 1504; 1507; 1515; 1521; 1522; 1532; 1552; 1553; 1555; 1556; 1562; 1563; 1565; 1571; 1573; 1585; 1587; 1592; 1597; 1599; 1600; 1605; 1612; 1632; 1636; 1665; 1670; 1671; 1681; 1686; 1689; 1693; 1696; 1698; 1706; 1713; 1718; 1720; 1725; 1728; 1730; 1732; 1740; 1741; 1752; 1759; 1760; 1762; 1765; 1768; 1769; 1770; 1788; 1810; 1828; 1831; 1835; 1837; 1839; 1849; 1851; 1853; 1861; 1863; 1871; 1872; 1873; 1875; 1885; 1889; 1892; 1896; 1898; 1901; 1903; 1905; 1910; 1914; 1921; 1936; 1937; 1947; 1951; 1956; 1962; 1964; 1968; 1970; 1982; 1983; 1984; 1986; 1991; 1992; 1994; 1999; 2011; 2020; 2022; 2025; 2028; 2031; 2034; 2039; 2054; 2059; 2069; 2084; 2085; 2087; 2089; 2094; 2101; 2105; 2116; 2119; 2127; 2132; 2137; 2138; 2139; 2140; 2153; 2155; 2158; 2162; 2167; 2176; 2187; 2197; 2209; 2218; 2220; 2221; 2222; 2227; 2228; 2230; 2231; 2234; 2238; 2245; 2248; 2251; 2252; 2257; 2258; 2264; 2265; 2270; 2274; 2284; 2297; 2300; 2304; 2309; 2320; 2325; 2330; 2336; 2342; 2346; 2350; 2354; 2358; 2360; 2361; 2367; 2381; 2382; 2383; 2384; 2390; 2393; 2397; 2405; 2406; 2410; 2425; 2433; 2437; 2438; 2442; 2446; 2452; 2453; 2463; 2468; 2469; 2484; 2491; 2493; 2494; 2496; 2497; 2500; 2511; 2514; 2515; 2523; 2528; 2530; 2537; 2538; 2540; 2543; 2550; 2552; 2554; 2558; 2563; 2570; 2571; 2585; 2596; 2601; 2608; 2616; 2619; 2621; 2624; 2627; 2629; 2644; 2648; 2650; 2651; 2656; 2659; 2663; 2669; 2670; 2687; 2690; 2691; 2693; 2696; 2699; 2700; 2703; 2704; 2708; 2712; 2734; 2745; 2749; 2750; 2753; 2754; 2762; 2766; 2777; 2780; 2783; 2789; 2792; 2794; 2802; 2805; 2809; 2818; 2823; 2826; 2827; 2829; 2834; 2836; 2837; 2841; 2845; 2851; 2852; 2862; 2874; 2878; 2883; 2885; 2895; 2901; 2914; 2915; 2916; 2922; 2925; 2929; 2930; 2935; 2937; 2938; 2939; 2949; 2953; 2959; 2960; 2962; 2965; 2966; 2969; 2970; 2978; 2984; 3001; 3007; 3019; 3020; 3023; 3025; 3032; 3043; 3058; 3070; 3072; 3075; 3090; 3093; 3098; 3100; 3103; 3117; 3128; 3134; 3144; 3148; 3156; 3158; 3159; 3162; 3165; 3167; 3169; 3176; 3181; 3182; 3183; 3189; 3191; 3195; 3196; 3202; 3203; 3208; 3217; 3219; 3222; 3223; 3231; 3235; 3244; 3256; 3261; 3267; 3282; 3284; 3286; 3292; 3295; 3298; 3312; 3314; 3321; 3322; 3324; 3328; 3335; 3348; 3352; 3358; 3364; 3369; 3377; 3383; 3384; 3389; 3394; 3399; 3400; 3406; 3410; 3418; 3419; 3422; 3427; 3428; 3431; 3435; 3446; 3447; 3449; 3461; 3468; 3470; 3481; 3488; 3489; 3491; 3492; 3494; 3501; 3527; 3539; 3543; 3547; 3557; 3560; 3582; 3588; 3594; 3600; 3601; 3603; 3604; 3607; 3616; 3625; 3627; 3631; 3634; 3635; 3650; 3655; 3660; 3665; 3666; 3672; 3681; 3683; 3688; 3691; 3711; 3713; 3714; 3715; 3721; 3723; 3725; 3727; 3736; 3737; 3740; 3747; 3749; 3751; 3755; 3757; 3764; 3765; 3767; 3768; 3772; 3783; 3788; 3793; 3813; 3816; 3817; 3818; 3821; 3823; 3827; 3832; 3844; 3845; 3846; 3853; 3854; 3858; 3860; 3866; 3867; 3868; 3869; 3870; 3873; 3887; 3888; 3891; 3896; 3897; 3898; 3900; 3901; 3915; 3916; 3921; 3932; 3942; 3947; 3950; 3957; 3959; 3960; 3963; 3965; 3977; 3978; 3981; 4003; 4013; 4015; 4023; 4026; 4029; 4031; 4034; 4038; 4065; 4075; 4082; 4096; 4098; 4100; 4107; 4110; 4111; 4113; 4114; 4119; 4120; 4124; 4127; 4133; 4134; 4135; 4146; 4150; 4156; 4157; 4182; 4187; 4192; 4198; 4208; 4212; 4216; 4217; 4220; 4222; 4225; 4226; 4233; 4243; 4244; 4246; 4252; 4255; 4257; 4259; 4260; 4266; 4269; 4281; 4286; 4287; 4290; 4292; 4296; 4308; 4314; 4315; 4331; 4337; 4345; 4346; 4348; 4356; 4366; 4373; 4376; 4377; 4379; 4385; 4390; 4399; 4401; 4414; 4416; 4419; 4428; 4429; 4430; 4432; 4447; 4464; 4479; 4481; 4482; 4483; 4492; 4495; 4497; 4510; 4512; 4521; 4526; 4532; 4534; 4535; 4538; 4545; 4555; 4563; 4565; 4566; 4574; 4580; 4596; 4597; 4603; 4615; 4619; 4621; 4624; 4625; 4630; 4642; 4645; 4651; 4652; 4654; 4656; 4659; 4665; 4667; 4668; 4678; 4685; 4686; 4689; 4691; 4694; 4695; 4699; 4703; 4713; 4719; 4724; 4729; 4754; 4755; 4761; 4765; 4776; 4777; 4778; 4784; 4787; 4791; 4792; 4797; 4798; 4800; 4801; 4804; 4811; 4813; 4815; 4829; 4836; 4846; 4873; 4877; 4880; 4882; 4885; 4888; 4896; 4898; 4908; 4930; 4933; 4934;

4946; 4948; 4951; 4952; 4954; 4961; 4964; 4969; 4979; 4990; 4995; 4996; 5010; 5011; 5013; 5014; 5015; 5020; 5023; 5024; 5038; 5041; 5042; 5043; 5044; 5045; 5051; 5053; 5057; 5058; 5065; 5066; 5067; 5079; 5084; 5088; 5090; 5093; 5094; 5095; 5098; 5118; 5123; 5128; 5139; 5143; 5144; 5146; 5147; 5154; 5156; 5160; 5162; 5164; 5171; 5174; 5175; 5184; 5186; 5188; 5190; 5196; 5197; 5198; 5200; 5220; 5223; 5228; 5230; 5239; 5245; 5246; 5251; 5256; 5261; 5276; 5277; 5280; 5284; 5286; 5295; 5304; 5307; 5308; 5310; 5311; 5317; 5323; 5329; 5351; 5354; 5364; 5368; 5372; 5384; 5410; 5417; 5419; 5420; 5422; 5423; 5430; 5441; 5443; 5450; 5462; 5464; 5465; 5476; 5480; 5483; 5484; 5489; 5492; 5493; 5501; 5505; 5511; 5512; 5517; 5522; 5527; 5528; 5534; 5542; 5547; 5553; 5559; 5563; 5576; 5583; 5587; 5589; 5590; 5597; 5599; 5613; 5616; 5619; 5625; 5627; 5629; 5644; 5650; 5655; 5667; 5676; 5680; 5688; 5689; 5692; 5694; 5703; 5708; 5709; 5715; 5716; 5720; 5722; 5728; 5729; 5730; 5735; 5742; 5744; 5746; 5749; 5752; 5753; 5755; 5763; 5767; 5777; 5783; 5794; 5808; 5811; 5813; 5818; 5821; 5836; 5839; 5841; 5853; 5868; 5879; 5886; 5888; 5891; 5892; 5904; 5910; 5927; 5933; 5941; 5942; 5945; 5946; 5949; 5951; 5952; 5958; 5962; 5972; 5973; 5974; 5976; 5981; 5982; 5986; 5989; 5995; 5996; 5997; 6010; 6016; 6026; 6028; 6033; 6037; 6049; 6059; 6060; 6066; 6070; 6074; 6075; 6077; 6082; 6084; 6085; 6088; 6093; 6102; 6106; 6109; 6117; 6119; 6122; 6128; 6131; 6134; 6145; 6149; 6157; 6158; 6161; 6162; 6166; 6177; 6199; 6203; 6207; 6208; 6211; 6212; 6215; 6218; 6220; 6221; 6224; 6241; 6242; 6248; 6252; 6255; 6260; 6261; 6263; 6267; 6282; 6284; 6287; 6290; 6292; 6295; 6303; 6310; 6315; 6326; 6340; 6341; 6348; 6353; 6357; 6359; 6361; 6368; 6371; 6373; 6374; 6381; 6383; 6388; 6396; 6397; 6400; 6411; 6414; 6415; 6417; 6418; 6420; 6427; 6442; 6443; 6448; 6458; 6459; 6465; 6469; 6473; 6474; 6478; 6486; 6501; 6505; 6506; 6514; 6518; 6521; 6527; 6528; 6529; 6530; 6544; 6550; 6551; 6553; 6557; 6561; 6565; 6569; 6575; 6581; 6586; 6591; 6602; 6607; 6611; 6614; 6627; 6637; 6641; 6646; 6647; 6653; 6654; 6659; 6660; 6663; 6666; 6673; 6677; 6680; 6681; 6683; 6684; 6689; 6690; 6691; 6693; 6702; 6714; 6724; 6735; 6744; 6749; 6753; 6756; 6758; 6771; 6772; 6773; 6774; 6779; 6784; 6792; 6795; 6806; 6810; 6817; 6822; 6824; 6826; 6827; 6842; 6852; 6857; 6872; 6877; 6880; 6883; 6886; 6889; 6891; 6900; 6912; 6917; 6919; 6920; 6925; 6927; 6928; 6929; 6941; 6943; 6947; 6949; 6955; 6960; 6964; 6974; 6975; 6990; 6997; 7001; 7006; 7008; 7010; 7013; 7015; 7019; 7022; 7026; 7036; 7038; 7039; 7040; 7048; 7050; 7058; 7060; 7062; 7072; 7074; 7076; 7080; 7083; 7101; 7123; 7141; 7142; 7143; 7146; 7149; 7151; 7152; 7159; 7170; 7171; 7179; 7181; 7183; 7186; 7191; 7193; 7198; 7205; 7213; 7215; 7218; 7222; 7225; 7230; 7240; 7241; 7246; 7275; 7279; 7299; 7306; 7311; 7312; 7314; 7319; 7324; 7326; 7338; 7340; 7346; 7348; 7349; 7355; 7356; 7358; 7359; 7379; 7389; 7390; 7396; 7404; 7407; 7412; 7417; 7418; 7423; 7435; 7438; 7439; 7444; 7452; 7459; 7473; 7475; 7479; 7481; 7482; 7484; 7488; 7489; 7491; 7492; 7500; 7507; 7517; 7529; 7533; 7538; 7540; 7541; 7542; 7545; 7556; 7565; 7568; 7569; 7570; 7584; 7585; 7590; 7591; 7592; 7597; 7604; 7606; 7611; 7612; 7614; 7615; 7620; 7621; 7622; 7625; 7629; 7639; 7640; 7650; 7655; 7656; 7662; 7670; 7673; 7674; 7684; 7687; 7689; 7691; 7702; 7710; 7711; 7713; 7715; 7717; 7730; 7744; 7747; 7748; 7755; 7762; 7766; 7778; 7783; 7788; 7795; 7803; 7804; 7807; 7817; 7822; 7824; 7834; 7837; 7838; 7846; 7848; 7850; 7856; 7857; 7858; 7859; 7863; 7872; 7880; 7881; 7886; 7887; 7921; 7928; 7932; 7936; 7937; 7939; 7944; 7946; 7950; 7953; 7957; 7964; 7968; 7969; 7970; 7976; 7977; 7989; 7993; 7996; 8007; 8010; 8011; 8014; 8021; 8038; 8039; 8044; 8054; 8061; 8065; 8070; 8073; 8074; 8079; 8080; 8083; 8088; 8090; 8097; 8100; 8109; 8112; 8117; 8122; 8124; 8125; 8129; 8136; 8143; 8153; 8156; 8165; 8171; 8182; 8198; 8202; 8213; 8216; 8233; 8234; 8235; 8236; 8237; 8247; 8249; 8250; 8255; 8257; 8259; 8262; 8273; 8276; 8279; 8286; 8287; 8290; 8294; 8298; 8315; 8321; 8327; 8331; 8335; 8336; 8340; 8345; 8348; 8349; 8354; 8356; 8367; 8376; 8378; 8382; 8390; 8404; 8406; 8413; 8415; 8419; 8420; 8422; 8427; 8437; 8438; 8443; 8454; 8464; 8469; 8471; 8473; 8476; 8478; 8479; 8481; 8487; 8502; 8503; 8508; 8511; 8513; 8522; 8523; 8528; 8535; 8539; 8542; 8543; 8548; 8552; 8553; 8556; 8560; 8581; 8586; 8587; 8593;

8597; 8605; 8606; 8611; 8612; 8620; 8622; 8629; 8632; 8634; 8635; 8648; 8649; 8655; 8668; 8685; 8686; 8693; 8703; 8704; 8719; 8726; 8742; 8745; 8747; 8748; 8753; 8755; 8762; 8765; 8766; 8767; 8775; 8787; 8788; 8790; 8794; 8801; 8803; 8805; 8809; 8811; 8814; 8817; 8818; 8819; 8830; 8836; 8842; 8847; 8859; 8863; 8870; 8872; 8875; 8877; 8880; 8884; 8886; 8888; 8895; 8898; 8899; 8902; 8907; 8908.

Codice C++

```cpp
#include <iostream>
#include <cmath>
#include <fstream>
#include <vector>

using namespace std;

long long mod_exp(long long base, long long exp, long long mod) {
    long long result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        exp = exp / 2;
        base = (base * base) % mod;
    }
    return result;
}

//restituisce true se probabilmente primo, false altrimenti
bool mcPrimalityTest(int n, int a){
    int s = 0;
    int q = n - 1;

    while(q % 2 == 0){
        s++;
        q = q / 2;
    }

    long long x = mod_exp(a, q, n);
    x = x % n;

    if(x == 1 || x == n - 1){
        return true;
    }

    while(s - 1 > 0){
        x = pow(x, 2);
        x = x % n;
        if (x == n - 1){
            return true;
```

```
        }

        s--;
    }

    return false;

}

int main() {

    vector<int> n = {561, 1105, 1729, 2465, 2821, 6601, 8911};
    vector<string> files = {"result561.txt", "result1105.txt", "result1729.txt",
"result2465.txt", "result2821.txt", "result6601.txt", "result8911.txt"};

    for(int i = 0; i < n.size(); i++){
        ofstream file(files[i]);
        for(int j = 2; j < n[i] - 1; j++){
            if(mcPrimalityTest(n[i], j)){
                file << j << "; ";
            }
        }
        file.close();
    }


    return 0;
}
```

Compito 9.1 Implementazione di MCMatrixMultiplicationVerifier

Ho implementato la funzione, successivamente per $k = 5, 10, 20$ ho ripetuto 100 volte l'aggiornamento di B e C, la sua perturbazione e poi l'esecuzione dell'algoritmo k volte, tenendo traccia di quante volte mi restituisse probabilmente uguali, se questo valore è uguale a k allora posso assumere ragionevolmente che $AB = C$ altrimenti, se mi restituisce anche solo una volta che sono diversi ho la certezza che $AB \neq C$.

Sappiamo che se $AB \neq C$, allora $P(A(Br) = Br) \leq \frac{1}{2}$. Cioè, la probabilità che dato AB è diverso da C MCMatrixMultiplicationVerifier dia che sono probabilmente uguali è inferiore a 0.5.

Ripetendo l'algoritmo k volte possiamo verificare $AB = C$ con probabilità $1 - \frac{1}{2^k}$.

Guardando i risultati ottenuti dall'implementazione dell'algoritmo possiamo confermare quanto descritto sopra:

- Con $k = 5$, seguendo quanto detto prima ci aspettiamo che l'algoritmo ci restituisca probabilmente uguale con probabilità $\frac{1}{2^5} = \frac{1}{32}$, quindi su 100 run ci aspettiamo che in circa 3 ci venga restituito che $AB = C$. Empiricamente, su 100 run, ho ottenuto che $AB = C$ per due volte su 100, rispettando così quello che ci aspetterebbe.
- Con $k = 10$, ci aspettiamo che l'algoritmo ci restituisca probabilmente uguale con probabilità $\frac{1}{2^{10}} = \frac{1}{1024}$, quindi su 100 run ci aspettiamo che in circa 0,1 ci venga restituito che $AB = C$, quindi altamente improbabile ma comunque possibile. Empiricamente, su 100 run, ho ottenuto che $AB = C$ per zero volte, in linea con le aspettative.
- Con $k = 20$, ci aspettiamo che l'algoritmo ci restituisca probabilmente uguale con probabilità $\frac{1}{2^{20}} = \frac{1}{1048576}$, quindi su 100 run ci aspettiamo che non ci venga restituito praticamente mai che $AB = C$, empiricamente, anche qua non ho mai ottenuto che $AB = C$. Per aspettarsi ragionevolmente di ottenere $AB = C$ per una volta, si dovrebbe eseguire l'algoritmo circa un milione di volte.

Codice C++

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <ctime>

#define DIM 10
#define RUNS 100

using namespace std;

int* prodottoMatriciVettore(int** mat, int n, int m, int* v, int lungVett) {
    if (n != lungVett) {
        cout << "Errore: dimensioni non compatibili" << endl;
        return nullptr;
    }
```

```
    int* r = new int[n];
    for (int i = 0; i < n; i++) {
        r[i] = 0;
        for (int j = 0; j < m; j++) {
            r[i] += mat[i][j] * v[j];
        }
    }
    return r;
}

int** prodottoMatriceMatrice(int** mat1, int n1, int m1, int** mat2, int n2, int
m2) {
    if (m1 != n2) {
        cout << "Errore: dimensioni non compatibili" << endl;
        return nullptr;
    }

    int** r = new int*[n1];
    for (int i = 0; i < n1; i++) {
        r[i] = new int[m2];
        for (int j = 0; j < m2; j++) {
            r[i][j] = 0;
            for (int k = 0; k < m1; k++) {
                r[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
    return r;
}

int** generaMatr(int n, int m) {
    int** mat = new int*[n];
    for (int i = 0; i < n; i++) {
        mat[i] = new int[m];
        for (int j = 0; j < m; j++) {
            mat[i][j] = rand() % 5 - 2;
        }
    }
    return mat;
}

//restituisce 1 se sono probabilmente uguali, 0 altrimenti
bool mcMatrixMultiplicationVerifier(int** A, int** B, int** C, int n) {
    int* r = new int[n];
    for (int i = 0; i < n; i++) {
        r[i] = rand() % 2;
    }
    int* s = prodottoMatriciVettore(B, n, n, r, n);
    int* t = prodottoMatriciVettore(A, n, n, s, n);
```

```
    int* u = prodottoMatriciVettore(C, n, n, r, n);

    //confronto t e u
    for (int i = 0; i < n; i++) {
        if (t[i] != u[i]) {
            return false;
        }
    }

    return true;
}

int main() {

    srand(time(NULL));

    vector<string> files = {"resultK5.txt", "resultK10.txt", "resultK20.txt"};
    vector<int> k = {5, 10, 20};

    int** matrA = generaMatr(DIM, DIM);

    for(int i = 0; i < k.size(); i++){

        ofstream file(files[i]);

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

            int** matrB = generaMatr(DIM, DIM);

            int** matrC = prodottoMatriceMatrice(matrA, DIM, DIM, matrB, DIM, DIM);

            matrC[rand() % DIM][rand() % DIM] += 1;

            int uguali = 0;
            int res;
            for (int z = 0; z < k[i]; z++) {
                if (res = mcMatrixMultiplicationVerifier(matrA, matrB, matrC, DIM))
{
                    uguali++;
                }
                //file << res << "; ";
            }
            //file << endl;
            int result = (uguali == k[i]) ? 1 : 0;
            file << result << endl;
        }
        file.close();
    }

    for (int i = 0; i < DIM; i++) {
```

```
        delete[] matrA[i];
    }
    delete[] matrA;

    return 0;
}
```

Compito 9.2

Ho usato come pool di numeri primi una lista pre-calcolata contenete i primi 400 numeri primi, successivamente ho applicato l'algoritmo MCStringEqualityVerifier a due file di 20bit (numeri binari) ed eseguito l'algoritmo campionando 20 primi diversi.

In nessuno dei casi ho ricevuto come output che le due stringhe sono probabilmente uguali, questo perché MCStringEqualityVerifier è un algoritmo monte carlo molto efficiente, la probabilità che restituisca uguale anche quando non lo sono è $P \approx \frac{2\ln(l)}{l}$.

Questo perché per a ≠ b, l'algoritmo restituisce probabilmente uguali quando la loro differenza modulo p è uguale a 0. |a-b| ha al più l fattori primi, inoltre, noi sappiamo dal teorema della numerosità dei numeri primi che il numero di questi minori di $l^2 = l^2/2\ln(l)$. Quindi la probabilità di errore è $\frac{l}{l^2/2\ln(l)} = \frac{2\ln(l)}{2}$. Questo ovviamente nel caso peggiore che |a-b| abbia l fattori primi. È chiaro come al crescere di l la probabilità che restituisca uguale diventa sempre più piccola.

Eseguendo l'algoritmo per 20 primi diversi non ho mai ottenuto che che le due stringe fossero uguali, un risultato aspettato in quanto:

Nel caso da me scelto i numeri binari sono:

- 10010110110010011101 = 617629
- 10010110100010011101 = 616605

La differenza tra i due è 1024, che ha un unico fattore primo 2 ($2^{10} = 1024$), quindi la probabilità che l'algoritmo restituisca uguale anche quando non lo sono è $\frac{1}{20^2/2\ln(20)} \approx 0{,}014$. Quindi eseguendo l'algoritmo un centinaio potremmo aspettare di ottenere per una run che sono uguali anche quando non lo sono (questo risultato è valido solo nel caso la differenza abbia un solo fattore primo, come il caso da me esposto).

Codice C++

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>

using namespace std;

vector<int> crivelloEratostene(int n){
    vector<int> primi;
    vector<bool> numeri(n+1, true);

    numeri[0] = numeri[1] = false;

    for(int i = 2; i <= n; i++){
        if(numeri[i]){
            for(int j = i * i; j <= n; j += i){
                numeri[j] = false;
            }
        }
```

```
    }
    for(int i = 2; i <= n; i++){
        if(numeri[i]){
            primi.push_back(i);
        }
    }

    return primi;
}
vector<int> primi = crivelloEratostene(400);

//restituisce true se probabilmente uguali, false se sono sicuramente diverse
bool mcStringEquality(int fileA, int fileB){

    int randomIndex = rand() % primi.size();
    int p = primi[randomIndex];

    int fA = fileA % p;
    int fB = fileB % p;

    if(fA == fB){
        return true;
    }

    return false;
}

int main() {

    srand(time(NULL));

    int fileA = 0b10010110110010011101;
    int fileB = 0b10010110100010011101;

    ofstream file("result.txt");

    for(int i = 0; i < 20; i++)
        file << mcStringEquality(fileA, fileB) << endl;

    cout << fileA << endl;
    cout << fileB << endl;
    cout << fileA - fileB << endl;
    file.close();

    return 0;
}
```