إثباتُ المُعجِزة العددية في حروف القُرآن المُقَطَّعة (حمٓ) و (عٓسٓقٓ)

باستخدام البرمجة

المهندس ماهر عمر أمين – المعهد التقني موصل– عراق

maherz55@yahoo.com

الملخص :
في سُوَر القُرآن الكريم الـ (114) توجد سَبْعُ سُوَر متتالية في ترتيبها مفتتحة بالحرفين (حم), ترتبط هذه السُوَر بعلاقات رياضية عجيبة مضى على اكتشافها حوالي عقدين من الزمن تربط بين مجموع تكرار ورود حَرفَي (الحاء والميم) في السُوَر السَبع وكمية توزيع الحرفين على السُوَر السَبع بطريقة رياضية مُحْكَمَة, في هذا البحث وباستخدام البرمجة بعد اختبار جميع احتمالات مزج الحروف وعددها أكثر من **(268) مليون** ترتيب, تبين أنه لا توجد أي مجموعة حروف (بأي **نوع** من الحروف وبأي **عدد** من الحروف) تستطيع تحقيق العلاقات التي تحقَّقت بالحَرفَين (الحاء والميم) وهذه النتيجة تقدم الدليل الحاسم على أن الحروف المُقَطَّعَة في سُوَر القرآن الكريم مرتبطة بعلاقات رياضية تحفظ القرآن من زيادة أي حرف أو نقصانه.
الكلمات الجوهرية: الحروف المُقَطَّعَة, حفظ القرآن من التغيير, توقيفية ترتيب سُوَر القرآن

1 المقدمة :

مِنَ المعلوم أن في القرآن الكريم تِسعاً وعِشرينَ سُورةً مُفتتحةً بحروف تُقرأ مُقَطَّعَةً بأسمائها ولم يَرِد من طريق صحيح عن النبي محمد (صلى الله عليه وسلم) بيانٌ للمراد منها وعلى الرغم من وجود أحاديث للرسول (صلى الله عليه وسلم) تَنهى وتُنذِر من يقول في القرآن بِرَأيِهِ فقد أثارت هذه الحروف الجدل الكثير, وتعددت فيها وجوه التفسير والتأويل, حتى تجاوزت بعض الآراء حدود المعقول, وتنكب بعضها عن جادة المنطق والعلم. وقد شُغِلَ بها المُفَسِرون, المُتَقَدِمُون منهم والمتأخِرون, كما شُغِلَ بها المُستشرقون(1) وأثارت عندهم الرغبة في استِكناهِهَا والتوصل إلى أسرارها, ممّا جعل بعض كتب التفسير مشحونةً بتأويلات وآراء في الحروف المُقَطَّعَة أوصَلَها بعضُهُم إلى عشرين رأياً(2-3), أغلبها بعيدة عن الفهم المعقول, وأما البحوث والدراسات(4-5) التي تناولتها فهي بالإضافة إلى أن أغلبها مُبتسَرة وغير منهجية لم تقدم حلاً معقولاً ومقنعاً(6).
وخلاصة القول أن جميع ما قيل في الحروف المُقَطَّعة عبارة عن آراء بشرية تُعَبِّر عن رَأي قائليها ليس فيها قولٌ واحد قد اعتمد على دليل. ومن هذه السُوَر سَبْعُ سُوَر متتالية في ترتيبها مفتتحة بالآية (حمٓ.) وهي (**غَافِر, فُصِّلَت, الشّورى, الزّخرُف, الدّخان, الجَاثِيَة, الأحْقاف**). وأرقام ترتيب هذه السُوَر في المصحف هي : (40 ، 41 ، 42 ، 43 ، 44 ، 45 ، 46).

وقد وجد بعض المعنيين بالأعداد في القرآن الكريم علاقات رياضية تسترعي الاهتمام تربط بين مجموع عدد الحَرفَين (الحاء والميم) في السُوَر السَبْع والعدد (19) من جهة ومجموع مكونات الأعداد للحَرفَين (الحاء والميم) في السُوَر السَبْع من جهة ثانية.
لقد بقيت العلاقات ولعقدين من الزمن منشورة على شبكة الانترنت ولا يستطيع أحدٌ أن يُثْبِتَ, أهِيَّ مُعجزة عددية أم ظاهرة رياضية اعتيادية؟ ولكي يتم التحقق بأسلوب علمي محايد بعيد عن أي تصورات مُسبَقة, كتبتُ برنامجاً حاسوبياً بلغة C++ لاختبار جميع احتمالات مزج الحروف بعضها مع بعض ليجيب عن سُؤال محدد: **هل يوجد ترتيب آخر من الحروف غير(الحاء والميم) يستطيع تحقيق مثل هذه العلاقات؟ وليخرج بنتائج إحصائية تحسم هذا الموضوع وتُؤصل لمنهجية جديدة في التعامل مع المعطَيات العددية المُكتَشَفة في القرآن الكريم.**

2 العلاقات الرياضية موضوع البحث :

الجدول-1 يبين علاقتين من سِتِّ علاقات تحقّقَ منها البحث واعتمد عليها وهي:

أ- إن مجموع تكرار الحرفين (الحاء والميم) في السُوَر السَبْع **يَقبلُ القِسمَة** على العدد **19** بدون باقٍ وناتج القسمة يساوي **113**. (لاحظ السطر الأخير في وسط الجدول)

ب- العجيب هو أن مجموع مُكوِّنات حاصل تجزئة الأعداد أفُقياً من العمودين السابقين يساوي **113** أيضاً, لاحظ مجموع مُكَوِّنات العمود في أقصى اليسار.

الجدول-1 يبين العلاقتين الأولى والثانية

A	السورة	رقمها	الافتتاحيات	ح	م	مُكونات العددين
1	غافر	40	حم.	64	380	3+8+0+6+4
2	فصلت	41	حم.	48	276	2+7+6+4+8
3	الشورى	42	حم. عسق.	53	300	3+0+0+5+3
4	الزخرف	43	حم.	44	324	3+2+4+4+4
5	الدخان	44	حم.	16	150	1+5+0+1+6
6	الجاثية	45	حم.	31	200	2+0+0+3+1
7	الأحقاف	46	حم.	36	225	2+2+5+3+6
			8	292	1855	113
					2147 = 19 X 113	

والجدول-2 يبين العلاقتين الثالثة والرابعة:

ج- السُور تنقسم إلى مجموعتين, ومجموع تكرار الحرفين (الحاء والميم) في كل مجموعة يقبل القسمة على **19**, المجموعة الأولى تضم السُوَر الثلاث الأوُلْ (3:2:1) والثانية تضم السُوَر الأربعة الأخيرة (7:6:5:4).

د- العجيب هو أن ظاهرة التساوي السابقة بين عددين من يمين ويسار الجدول تتكرر في قسمي (الجدول-2) , لاحظ ظهور العدد **59** مرتين في القسم **B** وظهور العدد **54** مرتين في **C**.

الجدول-2 يبين العلاقتين الثالثة والرابعة والانقسام الأول للسُوَر السَبْع إلى مجموعتين 4:5:6:7 | 1:2:3

B	السورة	رقمها	الافتتاحيات	ح	م	مُكونات العددين
1	غافر	40	حم.	64	380	3+8+0+6+4
2	فصلت	41	حم.	48	276	2+7+6+4+8
3	الشورى	42	حم.عسق.	53	300	3+0+0+5+3
			4	165	956	59
					1121 = 19 X 59	
C	السورة	رقمها	الافتتاحيات	ح	م	مُكونات العددين
4	الزخرف	43	حم.	44	324	3+2+4+4+4
5	الدخان	44	حم.	16	150	1+5+0+1+6
6	الجاثية	45	حم.	31	200	2+0+0+3+1
7	الأحقاف	46	حم.	36	225	2+2+5+3+6
			4	127	899	54
					1026 = 19 X 54	

والجدول-3 يبين العلاقتين الخامسة والسادسة:

هـ- تنقسم السُوَر السَبع مرة ثانية إلى مجموعتين, ومجموع تكرار الحرفين (الحاء والميم) في كل مجموعة يقبل القسمة على 19 . المجموعة الأولى تضم السُوَر (4:3:2) والمجموعة الثانية تضم السُوَر (7:6:5:1)

و- العجيب للمرة الثالثة هو ظهور حالة التساوي الغريبة بين الأعداد في يمين ويسار الجدول, لاحظ ظهور العدد 55 مرتين في القسم D وظهور العدد 58 مرتين في القسم E.

جدول-3 يبين العلاقتين الخامسة والسادسة والانقسام الثاني للسُوَر السَبْع إلى مجموعتين 1:5:6:7 | 2:3:4

D	السورة	رقمها	الافتتاحيات	ح	م	مكونات العددين
2	فصلت	41	حم.	48	276	2+7+6+4+8
3	الشورى	42	حم.عسق.	53	300	3+0+0+5+3
4	الزخرف	43	حم.	44	324	3+2+4+4+4
			4	145	900	55
				1045 = 19 X 55		
E	السورة	رقمها	الافتتاحيات	ح	م	مكونات العددين
1	غافر	40	حم.	64	380	3+8+0+6+4
5	الدخان	44	حم.	16	150	1+5+0+1+6
6	الجاثية	45	حم.	31	200	2+0+0+3+1
7	الأحقاف	46	حم.	36	225	2+2+5+3+6
			4	147	955	58
				1102 = 19 X 58		

ملاحظة : إحصائيات الحرفين أعلاه من ضمنها الحرفان (الحاء والميم) في (بسم الله الرحمن الرحيم) لكل سُوَرة.

إن علاقات القسمة على (19) بدون باقٍ أعلاه اكتُشِفَت في ثمانينيات القرن الماضي, أما علاقات (**تساوي عددين في طَرَفي كل جدول**) فقد اكتشفت على يد رجل جيكوسلوفاكي يُدعَى Milan Sulc في سنة 1993 اعتنق الإسلام وعُنِيَ بالعلاقات العددية في القرآن الكريم.

الجدير بالملاحظة أن جميع العلاقات السابقة تنهار بمجرد إضافة أو نقصان حرف من الحرفين (الحاء أو الميم), واضح أنها علاقات مقصودة تُحكم ربط السُوَر السَبْع بنظام رياضي مُحْكَم, وفي هذا البحث سيتم إثبات هذه الحقيقة (بالدليل المادي القاطع).

معلومات ذات صلة بالعلاقات السابقة : نعلم أن كل سُورة من السُور السَبْع تبدأ بقوله تعالى: **بسم الله الرحمن الرحيم* حمٓ ﴿1﴾**. والملاحظ أن السُوَرة الثالثة في المجموعة وهي سُورَة (الشُورَى) تأخذ وضعاً مختلفاً, فهي السُورَة الوحيدة في القرآن الكريم فيها آيتان, كلٌ منهما يتألف من الأحرف المُقَطَّعة وهي: **بسم الله الرحمن الرحيم* حمٓ ﴿1﴾ عٓسٓقٓ ﴿2﴾ إن ورود (عسق) في آية منفصلة في السورة الثالثة قد قسَّم آيات (حمٓ) في السُوَر السَبع إلى مجموعتين, المجموعة الأولى تضم السُوَر الثلاث الأولى (1:2:3) والمجموعة الثانية تضم السُوَر الأربع الأخيرة (4:5:6:7). لاحظ الجدول-2, وقد علمنا سابقاً من الجدول ذاته أن السُوَر السَبع تنقسم إلى نفس المجموعتين السابقتين (4:5:6:7 | 1:2:3) باعتبار آخر وهو أن مجموع الحَرفَين (الحاء والميم) في كل مجموعة يقبل القسمة على (19),** كذلك وأن (الجدول-3) يُقسِّم السُوَر السَبْع إلى مجموعتين (1:5:6:7 | 2:3:4) بنفس النسبة السابقة وهي 3 إلى 4.

ملاحظة : سأقتَصِر على العلاقات السابقة لِحَسْم الموضوع وفي الملحق أضعاف هذه العلاقات !

3 أهمية العدد (19) :

لنقرأ أولاً الآيات الكريمة في سُورَة (المدثر) : بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

﴿ لَوَّاحَةٌ لِّلْبَشَرِ ﴿29﴾ عَلَيْهَا تِسْعَةَ عَشَرَ ﴿30﴾ وَمَا جَعَلْنَآ أَصْحَٰبَ ٱلنَّارِ إِلَّا مَلَٰٓئِكَةً وَمَا جَعَلْنَا عِدَّتَهُمْ إِلَّا فِتْنَةً لِّلَّذِينَ كَفَرُوا لِيَسْتَيْقِنَ ٱلَّذِينَ أُوتُوا ٱلْكِتَٰبَ وَيَزْدَادَ ٱلَّذِينَ ءَامَنُوٓا إِيمَٰنًا وَلَا يَرْتَابَ ٱلَّذِينَ أُوتُوا ٱلْكِتَٰبَ وَٱلْمُؤْمِنُونَ وَلِيَقُولَ ٱلَّذِينَ فِى قُلُوبِهِم مَّرَضٌ وَٱلْكَٰفِرُونَ مَاذَآ أَرَادَ ٱللَّهُ بِهَٰذَا مَثَلًا كَذَٰلِكَ يُضِلُّ ٱللَّهُ مَن يَشَآءُ وَيَهْدِى مَن يَشَآءُ وَمَا يَعْلَمُ جُنُودَ رَبِّكَ إِلَّا هُوَ وَمَا هِىَ إِلَّا ذِكْرَىٰ لِلْبَشَرِ ﴿31﴾ ﴾

ذكر القرآن الكريم 30 عدداً مختلفاً وهي :

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 19, 20, 30, 40, 50, 60, 70, 80, 99, 100, 200, 300, 1000, 2000, 3000, 5000, 50000, & 100000.

ومجموع هذه الأعداد 162146 وهو من مضاعفات العدد (**19**) !

الجدير بالذكر أن جميع الأعداد المذكورة في القرآن الكريم ذُكِرَتْ لتحديد (**كَمِّية**), الاستثناء الوحيد هو العدد (**19**) الذي ذُكِر لتبيان (**خمسُ حِكَم إلهية**) من هذا العدد حددتها الآية الكريمة بالإضافة إلى تحديد (**كَمِّية**) وهي عدد خزنة جهنم, ومما يؤسف له أن كثيراً من المسلمين ومنهم ممن يُحسَبُون مِن العُلمَاء ما أن يُذكر العدد (**19**) حتى يربطوه بفرقة (**البهائية**) الضالة التي تُقَدِّس هذا العدد, وغَفِلُوا عن حقيقة مُهمَّة وهي أن الآية القرآنية الكريمة التي ذكرت الغايات من هذا العدد قد أُنزِلَت **قبل** ظهور البهائية بأكثر من إثني عشر قرناً ثم أن الحِكمَة الإلهية الأولى من هذا العدد وهي (**وَمَا جَعَلْنَا عِدَّتَهُمْ إِلَّا فِتْنَةً لِّلَّذِينَ كَفَرُوا**) قد تحقق فعلاً بظهور البهائية! وهي **معجزة قرآنية يجب أن يتنبه لها علماء المسلمين** وبالمناسبة نسأل هؤلاء: ماذا نفعل لو ظهرت فئةٌ تعبد (محمداً) صلى الله عليه وسلم؟ وهل نترك (محمداً) لوجود فئة ضالّة تعبُدُه؟

إن ما أكتشف من علاقات مرتبطة بالعدد (**19**) في القرآن الكريم (راجع الملحق) ينبيء عن منظومات عددية مُذهِلة ستكون من أهم وسائل الدعوة إلى الإسلام في المستقبل القريب بإذن الله.

4 منهجية البحث لاختبار العلاقات:

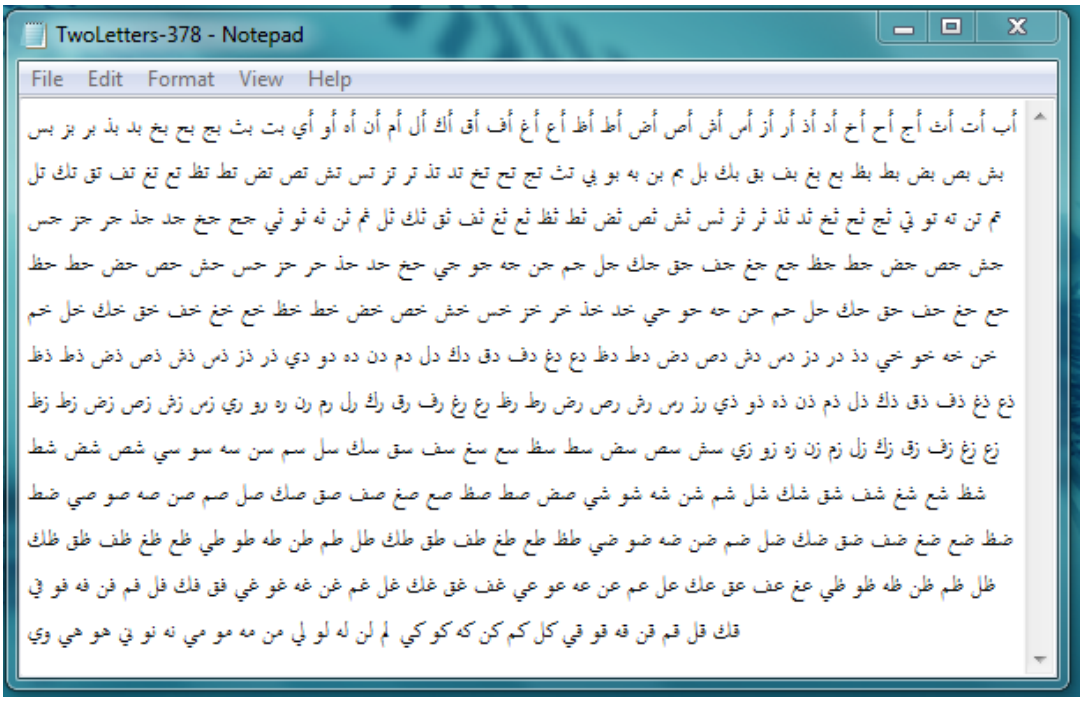
اختبار أولي : رأينا سابقاً ثلاث علاقات تقبل القسمة على العدد (**19**) وظهور خمسة أزواج من الأعداد المتساوية في يمين ويسار الجداول الثلاثة, والسؤال: ماذا لو استخدمنا عدداً آخر للقسمة غير العدد (19) فهل تظهر مثل هذه العلاقات؟ وما هو عدد العلاقات التي ستتحقق؟

للإجابة عن السؤال, تَمَّ تجربة الأعداد من **2** إلى **999** كعدد قاسم بدل العدد **(19)**.

والنتيجة هي أن العدد (**19**) هو الوحيد الذي يحقق جميع العلاقات, والعدد الوحيد الذي يستطيع تحقيق العلاقة الأولى فقط هو (113).

اختبار رئيسي : إن السُوَر السَبْع قد حققت العلاقات باستخدام الحَرفَين (الحاء والميم) حصراً, فهل يستطيع أيّ حرف أو مجموعة حروف للسُوَر السَبْع غير (**حم**) بأيّ عدد وبأيّ نوعية من الحروف: من حرف واحد وإلى **28** حرفاً تحقيق علاقات مشابهة؟

للإجابة عن السؤال, تم اختبار جميع مجاميع الحروف وعددها بالضبط (**268,435,455**).
وكما نعلم فإنه من البديهي في حالة اختبار (حرف واحد) وجود (**28**) طريقة لاختبار حروف اللغة العربية وتوجد (**378**) طريقة لاختبار كلِّ (حرفين) كما هو مبين في (الشكل-1) وتتزايد هذه المجاميع لتصل إلى (**40,116,600**) مجموعة عند مزج (14) حرفاً بعضها مع بعض, كما هو مبين في (الجدول-4) ثم تتناقص لتصل إلى (**مجموعة واحدة**) عند مزج (28) حرفاً,.



الشكل–1 يبين جميع احتمالات مزج حرفين مع بعض

الجدول – 4 يبين عدد مجاميع الحروف (من حرف واحد إلى 28 حرفاً)

عدد الحروف	عدد المجاميع	عدد الحروف	عدد المجاميع	عدد الحروف	عدد المجاميع	عدد الحروف	عدد المجاميع
1	28	8	3,108,105	15	37,442,160	22	376,740
2	378	9	6,906,900	16	30,421,755	23	98,280
3	3,276	10	13,123,110	17	21,474,180	24	20,475
4	20,475	11	21,474,180	18	13,123,110	25	3,276
5	98,280	12	30,421,755	19	6,906,900	26	378
6	376,740	13	37,442,160	20	3,108,105	27	28
7	1,184,040	14	40,116,600	21	1,184,040	28	1
المجموع الكلي لجميع احتمالات مزج الحروف الـ 28 يساوي 268,435,455							

5 خوارزمية عمل برنامج الحاسوب:

علمنا أن الغاية من البحث اختبار قابلية جميع احتمالات مزج الحروف على تحقيق علاقات شبيه بالعلاقات الستّ السابقة الملخصة في الجداول الثلاثة (3,2,1), ومن المفيد أن نتصور أن برنامج الحاسوب سيُكَوِّن (268,435,455) جدولاً شبيهاً بـ (الجدول-1) لكل مجموعة من الحروف المُختَبَرة وبعدد من الأعمدة الوسطية يتراوح بين 1 و 28 عموداً اعتماداً على عدد حروف كل مجموعة, ويختبر كل جدول إن كان يستطيع تحقيق العلاقة الأولى أو العلاقتين معاً ويسجل نتائج كل حالة, وهنا من المفيد أن نعلم بديهية رياضية وهي أنّ أيّ حالة مُختبرة تفشل في تحقيق العلاقة الأولى ستفشل حتماً في تحقيق انقسام السُوَر إلى قسمين وليس هناك حاجة لاختبار بقية

علاقات تلك الحالة. أمّا إذا وجد الحاسوب أيّ تركيب لحروف يحقق علاقات (الجدول-1) عندها فقط يختبر إن كان هذا التركيب يستطيع قِسمَة السُوَر إلى أقسام وبكل احتمالات الانقسام الممكنة (كما هو موضح في فقرة لاحقة) ويُخَزِّن تفاصيل الحالات التي تستطيع تحقيق أيّ حالة انقسام في ملفات دائمية ويقدم إحصائيات شاملة لكلّ الحالات المُختبرة.

ملاحظة : إن برنامجَ الحاسوب الذي يؤدي هذه المهمة مرفقٌ في نهاية البحث.

طرق الانقسام :

إن السُوَر السَبْع يمكن أن تنقسم إلى مجموعتين بـ **(63)** طريقة مختلفة مبينة في (الجدول-5) ولغرض التوضيح فقد صنفت الطرق في ثلاث مجموعات :

المجموعة A : تحافظ على الترتيب الأصلي في طرفي الانقسام, مثلاً 1:2:3 | 4:5:6:7

المجموعة B : تحافظ على الترتيب في أحد طرفي الانقسام فقط مع وجود قفزة في الترتيب في الطرف الآخر ويُستَبدَل كل عدد ناقص بالرمز (-) , مثلاً 2:3:4 | 1---5:6:7

المجموعة C : لا تحافظ على الترتيب التسلسلي للمجموعة الأصلية في كل من طرفي الانقسام.

الجدول-5 يوضح إحتمالية انقسام السُوَر السَبْع إلى قسمين بـ 63 طريقة مختلفة

	1	2	3	4	5
A	1|234567	12|34567	**123|4567**	1234|567	12345|67
6	123456|7				
B	2|1-34567	3|12-4567	4|123-567	5|1234-67	6|12345-7
13	23|1--4567	34|12--567	45|123--67	56|1234--7	**234|1---56**
	345|12---67	456|123---7	1-----7|23456		
C	1-3|2-4567	1--4|23-567	1---5|234-67	1----6|2345-7	2-4|1-3-567
	2--5|1-34-67	2---6|1-345-7	2----7|1-3456	3-5|12-4-67	3--6|12-45-7
	3---7|12-456	4-6|123-5-7	4--7|123-56	5-7|1234-6	12-4|3-567
	12--5|34-67	12---6|345-7	12----7|3456	1-34|2--567	1-3-5|2-4-67
44	1-3--6|2-45-7	1-3---7|2-456	1--45|23--67	1--4-6|23-5-7	1--4--7|23-56
	1---56|234--7	1---5-7|234-6	1----67|2345	23-5|1--4-67	23--6|1--45-7
	23---7|1--456	2-45|1-3--67	2-4-6|1-3-5-7	2-4--7|1-3-56	2--56|1-34--7
	2--5-7|1-34-6	2---67|1-345	34-6|12--5-7	34--7|12--56	3-56|12-4--7
	3-5-7|12-4-6	3--67|12-45	45-7|123--6	4-67|123-5	

6 البيانات التي اعتمد عليها البحث :

إن عدد حروف السُوَر السَبْع ولكل حرف من الحروف الـ (28) التي اعتمد عليها البرنامج مبينة في (الجدول-6). وهذه البيانات تم التأكد منها اعتماداً على ثلاثة إحصائيات, وهي مركز نون للدراسات القرآنية (7) و حسن محمد الجوهري (8) وعلي عبد الرزاق القره غولي (9) وهي متفقة في إحصاء جميع الحروف عدا اختلافها في حرف (الياء) فقط.

ملاحظة : إحصائيات الحروف أدناه من ضمنها حرفا (الحاء والميم) في **(بسم الله الرحمن الرحيم)** لكل سُوَرة.

الجدول-6 إحصائيات حروف السُوَر السَبْع التي اعتمد عليها البحث

الحروف		1 غافر	2 فصلت	3 الشورى	4 الزخرف	5 الدخان	6 الجاثية	7 الاحقاف
1	أ	816	568	533	531	234	323	444
2	ب	212	96	130	141	62	70	95
3	ت	142	98	82	112	46	95	91
4	ث	23	12	13	14	5	10	7
5	ج	41	29	32	44	15	19	25
6	ح	**64**	**48**	**53**	**44**	**16**	**31**	**36**
7	خ	33	25	15	32	7	16	16
8	د	115	68	64	63	15	21	57
9	ذ	91	56	57	55	19	30	48
10	ر	211	118	141	133	63	63	98
11	ز	23	23	22	11	10	16	13
12	س	100	65	54	72	23	40	48
13	ش	27	30	38	21	9	7	13
14	ص	33	19	28	17	4	9	20

الحروف		1 غافر	2 فصلت	3 الشورى	4 الزخرف	5 الدخان	6 الجاثية	7 الاحقاف
15	ض	26	16	23	24	4	13	18
16	ط	12	8	10	10	5	3	4
17	ظ	11	10	14	11	2	6	4
18	ع	142	102	98	121	47	60	79
19	غ	19	10	14	3	5	7	10
20	ف	149	92	84	94	32	43	64
21	ق	107	81	57	84	35	36	73
22	ك	187	86	93	112	50	68	77
23	ل	627	358	428	374	140	245	292
24	م	**380**	**276**	**300**	**324**	**150**	**200**	**225**
25	ن	404	296	257	330	155	151	216
26	هـ	229	173	194	209	67	111	128
27	و	373	252	276	285	117	158	206
28	ي	406	286	340	256	121	182	214

7 نتائج البحث :

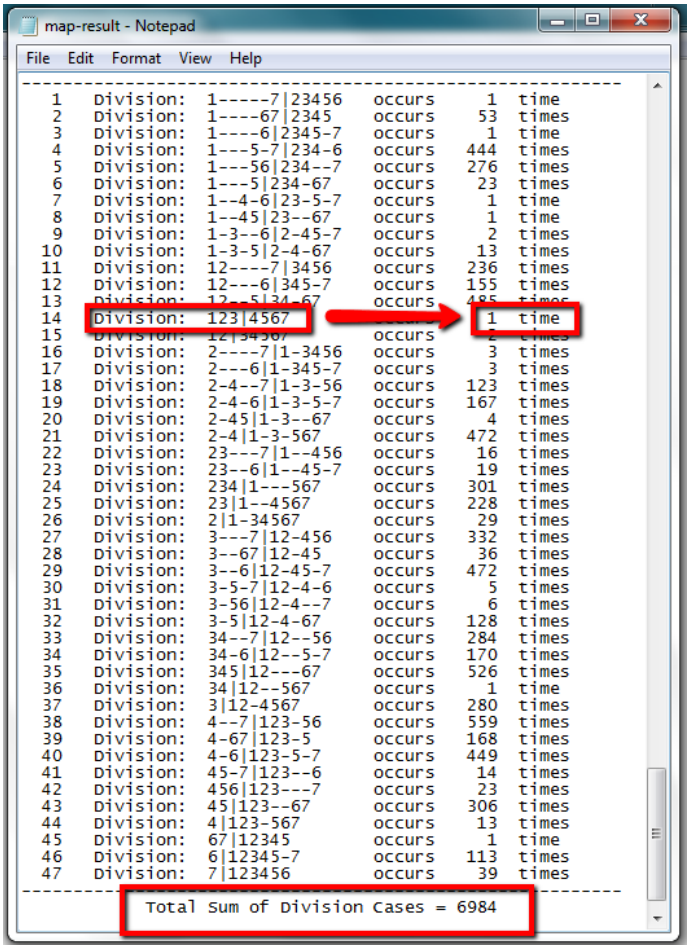
الجدول-7 يلخص نتائج اختبار أكثر من 268 مليون مجموعة مختلفة من الحروف

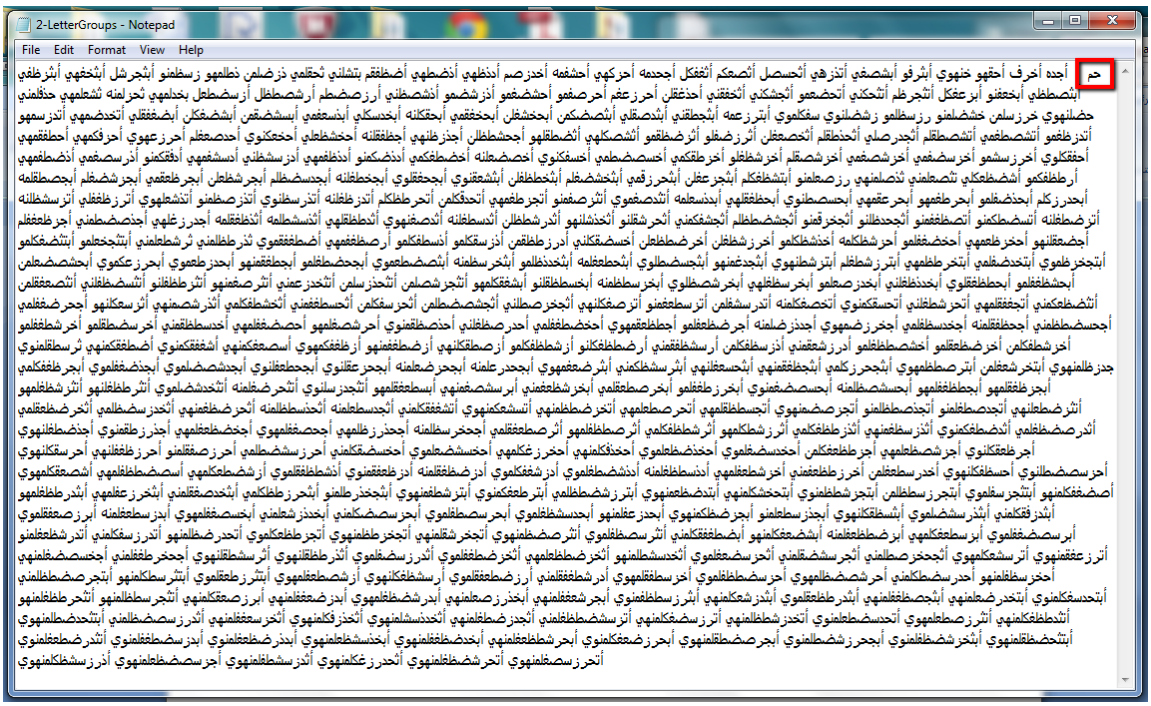
1	اختبر الحاسوب جميع احتمالات مزج الحروف في مجموعات وعددها 268,435,455.
2	26,761 حالة فقط حققت علاقات تشبه الجدول الأول أي بنسبة حدوث (1/10,000) !
3	5667 حالة فقط حققت انقسام واحد (الجدول الأول + جدول إنقسام)
4	444 حالة حققت انقسامين (كما حصل مع (حم)) أي بنسبة حدوث (1/600,000) !
5	128 حالة حققت ثلاثة انقسامات
6	5 حالات حققت أربعة انقسامات
7	5 حالات حققت خمسة انقسامات
8	مجموع حالات الانقسام = **6984** = 5667 + 444 x **2** + 128 x **3** + 5 x **4** + 5 x **5**

إن النتائج في الجدول-7 تقطع الطريق أمام أيَّ رَأيٍّ يقول (بالصُدفَة), والأعجب في التفاصيل :

- ✓ أن (**حم**) **فقط** من بين جميع حالات الانقسام وعددها (**6984**) حالة, تستطيع تحقيق الانقسام (1:2:3|4:5:6:7) الذي **يحافظ على نفس ترتيب السُوَر** في المصحف **ويتطابق كذلك مع الانقسام** الذي قسَّمَت به (**عسق**) السُوَر السَبْع ! لاحظ (الجدول-8).
- ✓ أن (حم) **فقط** (من بين 444 تركيب يحقق انقسامين) **تحتوي على أقل عدد من الحروف**, وأن (443) من مجاميع الحروف هذه غير صالح للاستخدام, لاحظ (الشكل-2)
- ✓ أن (حم) **فقط** (من بين 444 تركيب) **تحقق انقسامين بأقل عدد من الفجوات** بين السُوَر.

الجدول-8 التوزيع الإحصائي لجميع حالات الانقسام 6984 التي اكتشفها الحاسوب (كل نوع وتكرار حدوثه), لاحظ أن (حم) هو تركيب الحروف الوحيد الذي يحقق انقسام يحافظ على ترتيب السُوَر وبنفس قسمة (عسق) للسور





الشكل-2 يوضح مجاميع الحروف التي تحقق انقسامين

8 استنتاجات البحث

إن نتائجَ البحث ماديةٌ وليست قضيةً اجتهاديةً, فَلَمْ يَعُدْ هُناك مجالٌ للآراء, ونتائجُ البحث تحسمُ موضوعَ وجودِ المعجزات العددية في القرآن الكريم، وأقولها وبكل ثقة: لن يستطيع أحدٌ أن يأتي بمجموعةٍ من الحروف من الـ (268435455) الحالات التي اختبرها الحاسوب تستطيع أن تحقق ما حققه الحرفان (الحاء والميم) أو أن يأتي بنتائج تخالف ما توصل إليه برنامج الحاسوب من إحصائيات في هذا البحث اعتماداً على البيانات الموضحة في (الجدول-7) و (الجدول-8) ولو بزيادة أو نقصان رقم واحد!, وعلى الذين لا يستطيعون استيعابَ وجود معجزات عددية في القرآنِ الكريم بعد هذه النتائج المُذهِلة ويُصِّرونَ على معارضة الإعجاز العددي, أن يبحثوا عن العِلَّة في أسلوب تفكيرهم ومنهجَهُم الفكري.

الاستنتاجات :

- إن نتائج اختبار أكثر من (268) مليون مجموعة مختلفة من الحروف تُثبِت وبشكل قاطع أن الحروف المُقَطَّعَة قد وُضِعَت في بدايات السُوَر لتنبيه البشر في قابل الأيام على أن هناك معجزات عددية تربط بين هذه الحروف وأن على البشر أن يبحثوا عنها لِتكتَشَف في وقتها المعلوم الذي حدده الله بعلمِهِ السابق. وأن جميع ما قيل في الحروف المُقَطَّعَة سابقاً آراءٌ بشرية ليس من بينها قولٌ واحد قد اعتمد على دليل.
- تقديم **الدليل الرياضي المادي لغير المسلمين** الذي يثبت أنّ القرآنَ الكريم من عند الله, وأنه محفوظٌ كما نَزَلَ، وأن محمَدَاً رَسُولُ الله صِدقاً وَحَقَّاً. وأن الصحابة الكِرام ومن تَبِعَهُم قد حفظوا القرآن ولم يجتهدوا فيه ولم يبتَدِعُوا.
- أن هذه المعجزة العددية تقدِّم دليلاً رياضياً جديداً على **توقيفية ترتيب سُوَرِ القرآن الكريم**. و**حسم إحدى القضايا الخِلافيَّة بين المسلمين**.
- نتائج البحث تقدِّم الدليل الحاسم على **وجودِ أنظمةٍ عدديةٍ** في القرآن الكريم لإنهاء الجدل والنزاع حول موضوع الإعجاز العددي.
- إن **الإعجاز البلاغي** في القرآن الكريم على كمالِهِ سيُصبِحُ أكثرُ إعجازاً بتحقق المعجزات العددية والبلاغية في وقتٍ واحد.
- ولأن هذه المعجزة العددية قد اعتمدتْ على النظام العددي العَشري, لذا يُفهَم ضمناً أنَّ النظامَ العدديّ العشريّ ليس اختراعاً بشريّاً؛ بل هو **إلهامٌ إلهيٌ** للبشر.
- ويُستَنتَج كذلك أن الكشف عن المعجزات العددية يحتاج إلى استخدام البرمجة المتطورة.

9 التوصيات

- الاتصال مع منظمة العمل الإسلامي والمنظمات التابعة لها ومنها (الهيئة العالمية للإعجاز العلمي في القرآن والسنة) وكل مراكز البحوث الإسلامية لإطلاعها على النتائج.
- ترجمة البحث وإرساله إلى مراكز البحوث العلمية العالمية (غير الإسلامية على وجه الخصوص), لدراسة هذه الظاهرة العددية المُعجِزَة.
- تشجيع مراكز البحوث ذات الاهتمام بالموضوع لتجنيد باحثين لاستخدام لغات البرمجة المتقدمة في كشف عجائب القرآن الكريم العددية.
- المحافظة على برنامج البحث ونشره لأنه يقدم (**دليل مادي**) لهذه المعجزة العددية.

10 المراجع

(1) **نولدكه**: تاريخ القرآن - ط1- جوتنجن 1860
(2) **الرازي**: مفاتيح الغيب (= التفسير الكبير)
(3) **الزركشي:** البرهان في علوم القرآن.
(4) صبحي الصالح: مباحث في علوم القرآن- دار العلم للملايين بيروت ط8 1974.
(5) رمضان عبد التواب, حول فواتح بعض سور القرآن الكريم - من حوليات كلية الآداب جامعة عين شمس - العدد الثامن 1963
(6) عبد الجبار حمد حسين شرارة - الحروف المقطعة في القرآن الكريم - مجلة كلية الآداب - جامعة البصرة - العدد 16 1980.
(7) مركز نون للدراسات القرآنية www.islamnoon.com
(8) حسن محمد الجوهري www.quraananalysis7.net
(9) علي عبد الرزاق القره غولي www.heliwave.com

Unlock The Initialed Quranic Letters 'HaMeem' by Programming

Maher Omar Amin
Technical Institute- Mosul-Iraq
maherz55@yahoo.com

Abstract

The Quran is characterized by a unique phenomenon never found in any other book; 29 chapters are prefixed with 14 different sets of "Quranic Initials", consisting of one to five letters per set. Seven suras successively used in the Quran begin with the initial letters "Ha-Meem". these suras are the 40th to 46th suras. The total number of letters "Ha" and "Meem" is an exact multiple of 19. The code 19, is interlocked by a mathematical system with all the 7 suras show that these 7 suras are mathematically interlocked. In this research we wrote a computer program to examine all combinations for sets of letters from 1 to 28 to test if it satisfy the relations found in these 7 suras, exactly (268,435,455) cases were tested. The amazing result, there is no any case can satisfy the examined relations satisfied by (H.M).

Keywords: Initialed Quranic Letters, Quran is divinely preserved,

ملحق

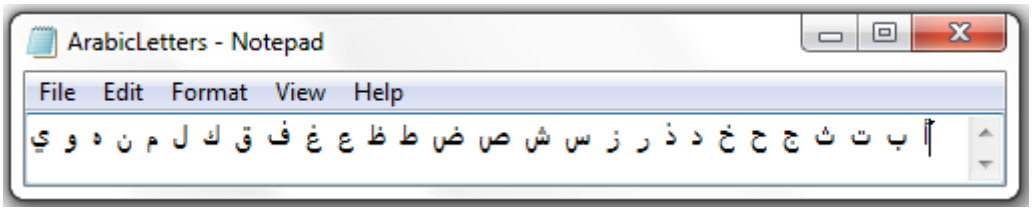
علاقات لـ (عسق) تكشف عن أبعاد مُستقبلية هائلة للمعجزات العددية

- ✓ عدد الحروف (ع س ق) في السورة المفتتحة بـ حم. عسق. يساوي (19x11) =209 ! وفيها كذلك
- ✓ عدد حروف (ع س) يساوي (19x8) =152 وعدد حروف (ق) يساوي (19x3) =57
- ✓ عدد الآيات التي تحتوي على الحروف (ع س ق) من السورة الأولى التي فيها (حم) وإلى الآية (عسق) يساوي (19x3) =57 !
- ✓ عدد الآيات التي تحتوي على الحروف (ع س ق) من الآية (عسق) في السورة الثالثة وإلى آخر آية من السورة السابعة في المجموعة يساوي (19x4) =76 !
- ✓ عدد الآيات التي تحتوي على الحروف (ع س ق) من الآية (عسق) وإلى نهاية المصحف الكريم يساوي (19x16)=304! آية.
- ✓ عدد الآيات في مجموعة السور السَبْع والتي يتساوى فيها أعداد حروف (ع س ق) يساوي (19x2)=38!
- ✓ عدد الآيات من السورة الأولى في المجموعة وإلى الآية (عسق) في السورة الثالثة يساوي 19 !
- ✓ نعلم إن الآية (عسق) تنتهي بالحرف (ق) , إن عدد الآيات من (عسق) وإلى نهاية المصحف الكريم والتي تنتهي بالحرف (ق) يساوي 19 !
- ✓ نعلم أن الآية (عسق) تبدأ بالحرف (ع) والآية التي تتحدث عن العدد (19) تبدأ بالحرف (ع) , إن عدد الآيات من (عسق) إلى الآية (عَلَيْهَا تِسْعَةَ عَشَرَ) وتبدأ بالحرف (ع) يساوي 19 !
- ✓ نعلم أن الآية التي تتحدث عن العدد (19) تنتهي بالحرف (ر). إن عدد الآيات في السور السبع وتنتهي بالحرف (ر) يساوي 19x2=38 !
- ✓ نعلم أن (عسق) هي الآية الوحيدة في القرآن الكريم التي فيها حروف مقطعة وتأخذ الرقم (2) أما بقية الآيات التي فيها حروف مقطعة وتتوزع على (29) سورة فتأخذ الرقم (1) , إن الآيات القرآنية التي تحمل الرقم (2) في كامل القرآن من السورة (1) وإلى السورة (114) تساوي 19 آية تحتوي على الحروف (ع س ق).
- ✓ إن آيات القرآن الكريم التي تحمل العدد (19) فيها 19 آية فقط تحتوي على الحروف (ع س ق).
- ✓ وفي هذه الآيات ألـ 19 يوجد (19x2) قاف !
- ✓ إن آيات القرآن الكريم التي تحمل الرقم (1) فيها 19 آية فقط تحتوي على الحروف (ع س ق).
- ✓ السورة المفتتحة بـ (حم.عسق) تحتوي على 19 آية فقط تحتوي على الحروف (ع س ق).
- ✓ وفي هذه الآيات ألـ 19 يوجد (19x2) قاف !
- ✓ إن آيات القرآن الكريم التي تحمل الرقم (1) ولا تحتوي هذه الآيات على كلمة (الله) فإن عدد حروف (ع س ق) في هذه الآيات يساوي (19x6)=114 وهو عدد سور القرآن الكريم !

برنامج الحاسوب

البرنامج مكتوب بلغة C++ , يتألف من ملف واحد, لتشغيل البرنامج تحتاج إلى تنصيب VC++ 2008 أو VC++ 2010 ونسخ ولصق محتويات الملف من هذه الصفحات, كما يجب توفر ملف بامتداد txt.

يحتوي على الحروف العربية الثمانية وعشرون كما في اللقطة المبينة أدناه :




```
/*
  PROGRAM: HaMeem.CPP
  Written by Maher Amin
  This program make 268,435,455 trials with all mixes of characters from 1 to 28 characters
  Last modified: 12/1/2014
*/
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <iterator>
#include <iomanip>
#include <direct.h>
#include <numeric>
#include <algorithm>
#include <ctime>
#include <map>
using namespace std;
const int HM_CHAPTERS = 7;
const int ARABIC_LETTERS = 28;
const int TwoDimData[ARABIC_LETTERS][HM_CHAPTERS] = {
        {816, 568, 533, 531, 234, 323, 444},     // 0 أ
        {212,  96, 130, 141,  62,  70,  95},     // 1 ب
        {142,  98,  82, 112,  46,  95,  91},     // 2 ت
        { 23,  12,  13,  14,   5,  10,   7},     // 3 ث
        { 41,  29,  32,  44,  15,  19,  25},     // 4 ج
        { 64,  48,  53,  44,  16,  31,  36},     // 5 ح
        { 33,  25,  15,  32,   7,  16,  16},     // 6 خ
        {115,  68,  64,  63,  15,  21,  57},     // 7 د
        { 91,  56,  57,  55,  19,  30,  48},     // 8 ذ
        {211, 118, 141, 133,  63,  63,  98},     // 9 ر
        { 23,  23,  22,  11,  10,  16,  13},     //10 ز
        {100,  65,  54,  72,  23,  40,  48},     //11 س
        { 27,  30,  38,  21,   9,   7,  13},     //12 ش
        { 33,  19,  28,  17,   4,   9,  20},     //13 ص
        { 26,  16,  23,  24,   4,  13,  18},     //14 ض
        { 12,   8,  10,  10,   5,   3,   4},     //15 ط
        { 11,  10,  14,  11,   2,   6,   4},     //16 ظ
        {142, 102,  98, 121,  47,  60,  79},     //17 ع
        { 19,  10,  14,   3,   5,   7,  10},     //18 غ
        {149,  92,  84,  94,  32,  43,  64},     //19 ف
        {107,  81,  57,  84,  35,  36,  73},     //20 ق
        {187,  86,  93, 112,  50,  68,  77},     //21 ك
        {627, 358, 428, 374, 140, 245, 292},     //22 ل
        {380, 276, 300, 324, 150, 200, 225},     //23 م
        {404, 296, 257, 330, 155, 151, 216},     //24 ن
        {229, 173, 194, 209,  67, 111, 128},     //25 هـ
        {373, 252, 276, 285, 117, 158, 206},     //26 و
        {406, 286, 340, 256, 121, 182, 214},     //27 ي
};

        void printArray( const unsigned int a[], const int size );
        void printVector( vector<unsigned int> vi, string str = "" );
```

```
        void print_vecOfvectors(vector<vector<unsigned int> > vvi);

        void printArray( const unsigned int a[], const int size )
        {       cout << "\n The array has " << size << " elements: {" << a[0];
                for (int i = 1; i <size; i++)  cout << ", " << a[i] ;
                cout << "}\n";
        }
        void printVector( vector<unsigned int> vi, string str  )
        {       cout << str;
                for (unsigned int i = 0; i <vi.size(); ++i)
                        cout << setfill(' ') << right << setw(5) << vi[i] ;
           cout << "\n";
        }

        void print_vecOfvectors(vector<vector<unsigned int> > vvi)
        {       cout << endl;
                for (unsigned int i=0; i<vvi.size(); ++i){
                        cout << endl;
                        for (unsigned int j=0; j<vvi[i].size(); ++j){
                                cout << setfill(' ') << right << setw(5) << vvi[i][j];
                        }
                cout << endl;
                }
                cout << endl;
        }
//-------------------------------------------------------------
class FinalHaMeem
{
public:
        FinalHaMeem();
        ~FinalHaMeem() { delete[]r1; delete[]r2; }
        void loop(unsigned int oneTo28);
        void mapResults();
private:
        enum {NUM_REL=150};
        enum {DIV=19};
        enum {LUA=25000};       // size of lookup array
        enum {MOD=80000};       // size of mod & isdivisor LookUp array
//-------------------------------------------------------------------
        int *r1, n1, k1;
        vector<unsigned int> vIndxComb1;
        int NextComb_n28(void);
        void makeComb_n28(int k);
//-------------------------------------------------------------------
        int *r2, n2, k2;
        vector<unsigned int> vIndxComb2;
        int NextComb_n7(void);
        void makeComb_n7(int k);
//-------------------------------------------------------------------
        vector<wchar_t> vArabicLetters;
        void read_ArabicLetters();
        unsigned int modArray[MOD];
        unsigned int divArray[MOD];
        unsigned int digitsLookUpArray[LUA];
        unsigned int sumDigits(unsigned int n);
        void fill_modArray();
        void fill_divArray();
        void fill_digitsLookUpArray();
        unsigned int freq[NUM_REL];
        void initializeResultArray();
        unsigned int TwoDimDgtData[ARABIC_LETTERS][HM_CHAPTERS];
        void fill_TwoDimDgtData();
        vector<vector<unsigned int> > vviData;
        vector<vector<unsigned int> > vviDgtData;
        unsigned int totalOfeachRow[ARABIC_LETTERS];
        unsigned int totalDigitsOfeachRow[ARABIC_LETTERS];
        void fill_totalAndDigitsOfeachRow();
        string mfolder, mSubfolder1, mSubfolder2;
        int caseScore;
        int caseDivisionScore;
        int caseTotalLetters;
        int caseTotalDigits;
        void calculateCaseLetterSumAndDigits();
        void printFreq(string fileName);
```

```
        int find_max_score();
        int combmax;
        vector<unsigned int> v1To7;
        void fill_v1To7();
        vector<vector<unsigned int> > vviInnerDivisions;
        void calculateCaseLetterSumAndDigitsPart1();
        int sumPart1;
        int sumDigitPart1;
        vector<unsigned int> viPart1, viPart2;
        void fill_part1();
        void fill_part2();
        void printLeftOrRight(string file, vector<unsigned int> v, string sep);
        void printCase(vector<unsigned int> vCharCombination, int option);
        void printSpecialCase(vector<unsigned int> vCharCombination);
        bool findSpecialCase();
        string itos(int i);
        void fill_vComplement(vector<unsigned int> vP1, vector<unsigned int>& vP2);
};
FinalHaMeem::FinalHaMeem()
{       n1 = ARABIC_LETTERS;
        r1 = new int[n1+1];
        n2 = HM_CHAPTERS;
        r2 = new int[n2+1];
        read_ArabicLetters();
        fill_modArray();
        fill_divArray();
        fill_digitsLookUpArray();
        fill_TwoDimDgtData();
        fill_totalAndDigitsOfeachRow();
        fill_v1To7();
}
void FinalHaMeem::read_ArabicLetters()
{       wifstream infile;
        infile.open ("ArabicLetters.txt");
        if (! infile) { cerr << "\nError while openinig file " << "ArabicLetters.txt" <<
"\n\n";
                exit(0);
        }
        wchar_t wc;
        int i = 0;
        while (infile >> wc) {
                vArabicLetters.push_back(wc);
                i++;
        }
        if (infile.eof() && i == ARABIC_LETTERS)
                infile.close();
        else if (infile.eof() && i < ARABIC_LETTERS) {
                infile.close();
                cerr << "\nNot enough data in file " << "ArabicLetters.txt" << "\n\n";
                exit(0);
        }
        else if (infile.eof() && i > ARABIC_LETTERS) {
                infile.close();
                cerr << "\nExcess data in file " << "ArabicLetters.txt" << "\n\n";
                exit(0);
        }
        else {infile.close();
                        cerr << "\nError in data inside file " << "ArabicLetters.txt" <<
"\n\n";
                        exit(0);
        }
        infile.close();
}
void FinalHaMeem::fill_v1To7()
{       v1To7.clear();
        for (unsigned int i=0; i<HM_CHAPTERS; ++i)
                v1To7.push_back(i+1);
}
unsigned int FinalHaMeem::sumDigits(unsigned int n)
{       unsigned int digitSum = 0;
        do {digitSum = digitSum + n%10;
                n = n/10;
        } while (n != 0);
        return digitSum;
```

```
}
void FinalHaMeem::fill_modArray()
{       for (unsigned int i=0; i<MOD; ++i){
                if (i % DIV == 0)
                        modArray[i] = 1;
                else modArray[i] = 0;}
}
void FinalHaMeem::fill_divArray()
{       for (unsigned int i=0; i<MOD; ++i)
                divArray[i] = i/DIV;
}
void FinalHaMeem::fill_digitsLookUpArray()
{       for (unsigned int i=0; i<LUA; ++i)
                digitsLookUpArray[i] = sumDigits(i);
}
void FinalHaMeem::initializeResultArray()
{       for (int i=0; i<NUM_REL; ++i)
                freq[i] = 0;
}
void FinalHaMeem::fill_TwoDimDgtData()
{       for (unsigned int j=0; j<ARABIC_LETTERS; ++j)
                for (unsigned int k=0; k<HM_CHAPTERS; ++k)
                        TwoDimDgtData[j][k] = digitsLookUpArray[TwoDimData[j][k]];
}
void FinalHaMeem::fill_totalAndDigitsOfeachRow()
{       for (unsigned int i=0; i< ARABIC_LETTERS; ++i){
                int sum = 0, digits = 0;
                for (unsigned int j=0; j<HM_CHAPTERS; ++j){
                        sum = sum + TwoDimData[i][j];
                        digits = digits + TwoDimDgtData[i][j];
                }
                totalOfeachRow[i] = sum;
                totalDigitsOfeachRow[i] = digits;
        }
}
void FinalHaMeem::loop(unsigned int oneTo28)
{       clock_t begin = clock();
//-------------------------------------------------------------------
        mfolder     = "HaMeemRESULT";
        mSubfolder1 = "HaMeemRESULT/DIVISIONS";
        mSubfolder2 = "HaMeemRESULT/LETTER-GROUPS";
        _mkdir(mfolder.c_str());
        _mkdir(mSubfolder1.c_str());
        _mkdir(mSubfolder2.c_str());
//-------------------------------------------------------------------
        combmax = oneTo28;
        makeComb_n28(combmax);
        clock_t end = clock();
        double elapsed_secs = double(end - begin) / CLOCKS_PER_SEC;
        cout << "\nElapsed time = " << elapsed_secs << endl;
}       // end of loop function
void FinalHaMeem::makeComb_n28(int numLetters)
{       initializeResultArray();
        r1[0] = 0;
        for (int i=1; i<=n1; i++)
                r1[i] = i;
//---------------------------
        k1 = numLetters;
        int i, j=0;
        do {
                vIndxComb1.clear();
                for (i=1; i<=numLetters; i++){
                        vIndxComb1.push_back(r1[i]-1);
                }
                //---------------------------------------
                calculateCaseLetterSumAndDigits();
                caseScore = 0;
                if ( modArray[caseTotalLetters]  && caseTotalLetters != 0 )
                        caseScore++;
                else { freq[caseScore]++;
                continue;
                }
                if (divArray[caseTotalLetters] == caseTotalDigits)
                        caseScore++;
```

```
                else { freq[caseScore]++;
                continue;
                }
                caseDivisionScore = 0;
                vviInnerDivisions.clear();

                for (unsigned int i=1; i<=(HM_CHAPTERS)/2; ++i)
                        makeComb_n7(i);
                if (caseDivisionScore > 0) {
                        caseScore = caseScore + 2*caseDivisionScore;
                        freq[caseScore]++;
                        printCase(vIndxComb1, 1);
                        if (findSpecialCase() && caseDivisionScore > 0)
                                printSpecialCase(vIndxComb1);
                }
                else freq[caseScore]++;
        } while (NextComb_n28());
        printFreq("Statistics");
}
void FinalHaMeem::makeComb_n7(int numLetters)
{       r2[0] = 0;
        for (int i=1; i<=n2; i++)
                r2[i] = i;
//----------------------------
        k2 = numLetters;
        int i, j=0;
        do {
                vIndxComb2.clear();
                for (i=1; i<=numLetters; i++){
                        vIndxComb2.push_back(r2[i]-1);
                }
                //---------------------------------------
                calculateCaseLetterSumAndDigitsPart1();
                if (sumPart1 % DIV == 0 && sumPart1/DIV == sumDigitPart1 && sumPart1 != 0){
                        caseDivisionScore++;
                        fill_part1();
                        fill_part2();
                        vviInnerDivisions.push_back(vector<unsigned int>());
                        for (unsigned int i=0; i<viPart1.size(); ++i)
                                vviInnerDivisions[vviInnerDivisions.size()-
                        1].push_back(viPart1[i]);
                }
        } while (NextComb_n7());
}
int FinalHaMeem::NextComb_n28()
{       int i = k1,j;
        while (i > 0 && r1[i] == n1-k1+i)
                i--;
        if (i == 0)
                return 0;
        r1[i]++;
        for (j=i+1; j<=k1; j++)
                r1[j] = r1[j-1] + 1;
        return 1;
}
int FinalHaMeem::NextComb_n7()
{       int i = k2,j;
        while (i > 0 && r2[i] == n2-k2+i)
                i--;
        if (i == 0)
                return 0;
        r2[i]++;
        for (j=i+1; j<=k2; j++)
                r2[j] = r2[j-1] + 1;
        return 1;
}
void FinalHaMeem::calculateCaseLetterSumAndDigits()
{       caseTotalLetters = 0;
        caseTotalDigits  = 0;
        for (unsigned int i=0; i<vIndxComb1.size(); ++i){
                caseTotalLetters = caseTotalLetters + totalOfeachRow[vIndxComb1[i]];
                caseTotalDigits  = caseTotalDigits  + totalDigitsOfeachRow[vIndxComb1[i]];
        }
}
```

```
void FinalHaMeem::calculateCaseLetterSumAndDigitsPart1()
{       sumPart1 = 0;
        sumDigitPart1 = 0;
        for (unsigned int i=0; i<vIndxComb1.size(); ++i)
        for (unsigned int j=0; j<vIndxComb2.size(); ++j){
                sumPart1 = sumPart1 + TwoDimData[vIndxComb1[i]][vIndxComb2[j]];
                sumDigitPart1 = sumDigitPart1 + TwoDimDgtData[vIndxComb1[i]][vIndxComb2[j]];
        }
}
void FinalHaMeem::fill_part1()
{       viPart1.clear();
        for (unsigned int i=0; i<vIndxComb2.size(); ++i)
                viPart1.push_back(vIndxComb2[i]+1);
}
void FinalHaMeem::fill_part2()
{       viPart2.clear();
        set_difference (v1To7.begin(), v1To7.end(), viPart1.begin(), viPart1.end(),
back_inserter(viPart2));
}
string FinalHaMeem::itos(int i)
{       stringstream s;
        s << i;
        return s.str();
}
void FinalHaMeem::fill_vComplement(vector<unsigned int> vP1, vector<unsigned int>& vP2)
{       vP2.clear();
        vP2.reserve(HM_CHAPTERS-viPart1.size());
        set_difference (v1To7.begin(), v1To7.end(), vP1.begin(), vP1.end(),
back_inserter(vP2));
}
void FinalHaMeem::printCase(vector<unsigned int> vCharCombination, int option)
{
        string pre = itos(vviInnerDivisions.size());
        string fileName1 = mSubfolder1 + "/" + pre + "-DivisionCases.txt";
        string fileName2 = mSubfolder2 + "/" + pre + "-LetterGroups.txt";
        string fileName3 = mfolder + "/All-DivisionCases.txt";
        string fileName4 = mSubfolder1 + "/" + pre + "-abstractDivisionCases.txt";
        wofstream outfile1( fileName1.c_str(), ios_base::app );
        wofstream outfile2( fileName2.c_str(), ios_base::app );
        wofstream outfile3( fileName3.c_str(), ios_base::app );
        wofstream outfile4( fileName4.c_str(), ios_base::app );
        outfile1 << endl;
        outfile3 << endl;
        for (unsigned int i=0; i<vCharCombination.size(); ++i){
                outfile1 << vArabicLetters[vCharCombination[i]];
                outfile2 << vArabicLetters[vCharCombination[i]];
                outfile3 << vArabicLetters[vCharCombination[i]];
        }
        outfile1 << endl;
        outfile2 << " ";
        outfile3 << endl;
        for (unsigned int i=0; i<vIndxComb1.size(); ++i){
                outfile1 << vIndxComb1[i]+1 << " ";
                outfile3 << vIndxComb1[i]+1 << " ";
        }
        outfile1 << endl;
        outfile3 << endl;
        outfile1 << vviInnerDivisions.size() << endl;
        outfile3 << vviInnerDivisions.size() << endl;
        for (unsigned int j=0; j<vviInnerDivisions.size(); ++j){
                vector<unsigned int> vLeft;
                vector<unsigned int> vRight;
                for (unsigned int n=0; n<vviInnerDivisions[j].size(); ++n)
                        vLeft.push_back(vviInnerDivisions[j][n]);
                fill_vComplement(vLeft, vRight);
                printLeftOrRight(fileName1, vLeft, "|");
                printLeftOrRight(fileName1, vRight, "");
                printLeftOrRight(fileName3, vLeft, "|");
                printLeftOrRight(fileName3, vRight, "");
                printLeftOrRight(fileName4, vLeft, "|");
                printLeftOrRight(fileName4, vRight, "");
                outfile4 << " ";
                outfile1 << endl;
                outfile3 << endl;
```

```
        }
        outfile4 << endl;
        outfile4.close();
        outfile1.close();
        outfile2.close();
        outfile3.close();
}
void FinalHaMeem::printSpecialCase(vector<unsigned int> vCharCombination)
{       int tmp = vviInnerDivisions.size();
        string pre = itos(tmp);
        string fileName1 = mfolder + "/SpecialCases.txt";
        wofstream outfile1( fileName1.c_str(), ios_base::app );
        outfile1 << endl;
        for (unsigned int i=0; i<vCharCombination.size(); ++i)
                outfile1 << vArabicLetters[vCharCombination[i]];
        outfile1 << endl;
        for (unsigned int i=0; i<vIndxComb1.size(); ++i)
                outfile1 << vIndxComb1[i]+1 << " ";
        outfile1 << endl;
        outfile1 << vviInnerDivisions.size() << endl;
        for (unsigned int j=0; j<vviInnerDivisions.size(); ++j){
                vector<unsigned int> vLeft;
                vector<unsigned int> vRight;
                for (unsigned int n=0; n<vviInnerDivisions[j].size(); ++n)
                        vLeft.push_back(vviInnerDivisions[j][n]);
                fill_vComplement(vLeft, vRight);
                printLeftOrRight(fileName1, vLeft, "|");
                printLeftOrRight(fileName1, vRight, "");
                outfile1 << endl;
        }
        outfile1.close();
}
bool FinalHaMeem::findSpecialCase()
{       vector<unsigned int> vtmp;
        vtmp.push_back(1);
        vtmp.push_back(2);
        vtmp.push_back(3);
        bool satisfy = false;
        for (unsigned int i=0; i<vviInnerDivisions.size(); ++i)
            if (vviInnerDivisions[i] == vtmp) satisfy = true;
        return satisfy;
}
void FinalHaMeem::printLeftOrRight(string file, vector<unsigned int> v, string sep)
{       wofstream outfile( file.c_str(), ios_base::app );
        for (unsigned int k=0; k<v.size(); ++k){
                if (k == 0)
                        outfile << v[k];
                        else if (v[k] == v[k-1]+1)
                                outfile << v[k];
                        else if (v[k] > v[k-1]+1){
                                int tm = v[k] -1- v[k-1];
                                for (int r=0; r<tm; ++r)
                                    outfile << "-";
                                outfile << v[k];
                        }
        }
        outfile << sep.c_str();
        if (sep == "") outfile << "  ";
        outfile.close();
}
int FinalHaMeem::find_max_score()
{       int i, max = NUM_REL-1;
        for (i = max; i>0; --i )
                if( freq[i] != 0 )
                   return i;
    return i;
}
void FinalHaMeem::printFreq(string fileName)
{
        int width = 15;
        int max = find_max_score();
        fileName = fileName + ".txt";
        string file = mfolder + "/" + fileName;
        ofstream outfile( file.c_str(), ios_base::app );
```

```
        string s1(58, '-');
        outfile << s1 << endl;
        cout << s1 << endl;

        outfile << "\nResult for " << combmax << " letters, with total of "
                    << accumulate(freq, freq+NUM_REL, 0) << " cases tested. \n";
        cout <<    "\nResult for " << combmax << " letters "
                  << accumulate(freq, freq+NUM_REL, 0) << " cases tested. \n";
        outfile << s1 << endl;
        cout <<    s1 << endl;
        for (int i = max; i >= 0; --i) {
   outfile << setw(28) << "  Number of cases in store[ " << right << setw(2) << i << " ] =
outfile << setw(width) << freq[i] << "\n\n";        cout <<    setw(28) << "  Number of cases
in store[ " << right << setw(2) << i << " ] = ";
           cout <<    setw(width) << freq[i] << "\n\n";
        }
        outfile << s1 << endl;
        outfile.close();
        cout << s1 << endl;
}
void FinalHaMeem::mapResults()
{       string fileName = mfolder + "/All-DivisionCases.txt";
// count number of times each word occurs in the input
        map<string, int> word_count;
        ifstream myfile (fileName.c_str(), ios::in);
        if ( myfile.is_open() ){
                string st;
        while ( myfile >> st )
        {
                for (unsigned int i=0; i<st.size(); ++i){
                        if (st[i] == '|'){
                           ++word_count[st];
                           break;
                        } }
        }
        myfile.close();
                cout << endl;
                int sum = 0;
                string s1(58, '-');
                cout << s1 << endl;
                // print the results
                int cn = 1;
                string tmp = "xxxxxxxxxxxxxx";
                string::size_type maxlen = tmp.size();
                for (map<string, int>::const_iterator it = word_count.begin();
                        it != word_count.end(); ++it) {
                        cout << setfill(' ') << right << setw(4) << cn << "   Division:  ";
                        cout << it->first ;
                        cout << string(maxlen - (it->first.size()), ' ') << " occurs ";
                        cout << setfill(' ') << right << setw(5) << it->second;
                        cout << ((it->second > 1) ? "  times" : "  time") << endl;
                        sum = sum + it->second;
                        cn++;
                }
                cout << s1 << endl;
                cout << "            Total Sum of Division Cases = " << sum << endl << endl;
        }
}
int main() {
    FinalHaMeem obj;
    for (unsigned int i=1; i<=28; ++i)
    obj.loop(i);
    obj.mapResults();
    getchar();
    return 0;
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```