Лабораторна робота №4.
Тема: Машина Тьюринга. Нормальний алгоритм Маркова.

Питання для опрацювання.
1. Скінченні автомати.
2. Автомати з машинною пам'яттю.
3. Машини Тьюрінга.
4. Автомати Маркова.
5. Машина Поста.

Приклади складання програм для МТ

Розглянемо приклади складання програм для МТ, щоб продемонструвати деякі типові прийоми програмування на МТ.

Для скорочення формулювання задач введемо наступні дві домовленості:

– літерою *P* будемо позначати вхідне слово;

– літерою *A* будемо позначати алфавіт вхідного слова, тобто набір тих символів, з яких і тільки яких може складатися *P* (зазначимо, однак, що в проміжкових і вихідному словах можуть з`являтися і інші символи).

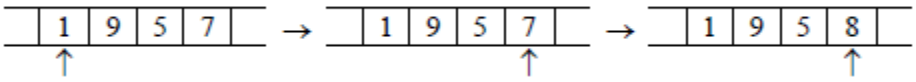
Приклад 1 (переміщення автомату, заміна символів)

A={0,1,2,3,4,5,6,7,8,9}. Пусть *P*– непорожнє слово; значить, *P* – це послідовність з десяткових цифр, тобто запис невід`ємного цілого числа в десятковій системі. Треба отримати на стрічці запис числа, яке на 1 більше за число *P*.

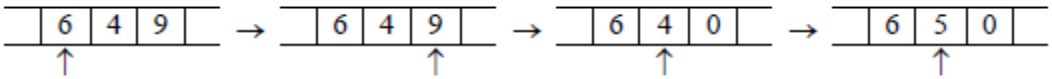
Рішення.

Для вирішення цієї задачі пропонується виконати наступні дії:

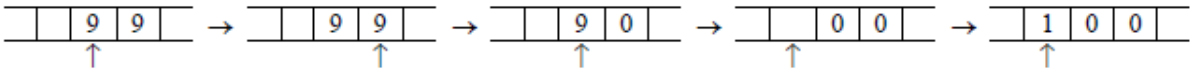
1. Перегнати автомат під останню цифру числа.
2. Якщо це цифра від 0 до 8, то замінити її цифрою на 1 більшою і зупинитися; наприклад:



3. Якщо ж це цифра 9, тоді замінитиїї на 0 і зсунути автомат до попередньої цифри, після чого таким самим способом збільшити на 1 цю передостанню цифру; наприклад:



4. Особливий випадок: в *P* тільки дев`ятки (наприклад, 99). Тоді автомат буде зсуватися вліво, замінюючи дев`ятки на нулі, і в кінці кінців виявиться під порожньою клітинкою. В цю порожню клітинку треба записати 1 ізупинитися (відповіддю буде 100):



У вигляді програми для МТ ці дії описуються наступним чином:

	0	1	2	3	4	5	6	7	8	9	Λ
q1	0,R,q1	1,R,q1	2,R,q1	3,R,q1	4,R,q1	5,R,q1	6,R,q1	7,R,q1	8,R,q1	9,R,q1	Λ,L,q2
q2	1,N,!	2, N,!	3, N,!	4, N,!	5, N,!	6, N,!	7, N,!	8, N,!	9, N,!	0,L,q2	1,N,!

Пояснення.

q1 – це стан, в якому автомат «біжить» під останню цифру числа. Для цього він весь час рухається вправо, не змінюючи видимі цифри і залишаючись в тому ж стані. Але тут є одна особливість: коли автомат знаходиться під останньою цифрою, то він ще не знає про це (бо він не бачить, що записано в сусідніх клітинках) і визначить це лише тоді, коли

попаде на порожню клітинку. Тому, підійшовши до першої порожньої клітинки, автомат повертається назад під останню цифру і переходить у стан *q2* (вправо рухатися вже не треба).

q2 – це стан, в якому автомат додає 1 до тієї цифри, яку бачить в даний момент. Спочатку це остання цифра числа; якщо вона – в діапазоні від 0 до 8, то автомат замінює її цифрою, яка на 1 більше, і зупиняється. Але якщо це цифра 9, то автомат замінює її на 0 і зсувається вліво, залишаючись в стані *q2*. Тим самим, він буде тепер додавати 1 до попередньої цифри. Якщо і ця цифра дорівнює 9, то автомат замінює її на 0 і зсувається вліво, залишаючись все ще в стані *q2*, т.як повинен виконати ту ж саму дію – збільшувати на 1 видиму цифру. Якщо ж автомат зсунувся вліво, а у видимій клітинці немає цифри (а є «порожньо»), то він записує сюда 1 ізупиняється. Заначимо, що для порожнього вхідного слова наша задача не визначена,тому на цьому слові МТ може вести себе як завгодно. В нашій програмі, наприклад, при порожньому вхідному слові МТ зупиняється і видає відповідь 1.

Вище ми навели запис програми в повному, нескороченому вигляді.Тепер же приведемо запис програми в скороченому, більш наочному вигляді, тому справа надамо коротке пояснення дій, які реалізуються у відподних станах автомату:

	0	1	2	3	4	5	6	7	8	9	Λ	
q1	,R,	,R,	,R,	,R,	,R,	,R,	,R,	,R,	,R,	,R,	,L,q2	под последнюю цифру
q2	1, ,!	2, ,!	3, ,!	4, ,!	5, ,!	6, ,!	7, ,!	8, ,!	9, ,!	0,L,	1, ,!	видимая цифра + 1

Саме так ми і будемо в подальшому записувати програми.

Приклад 2 (аналіз символів)

A={*a,b,c*}. Перенести перший символ непорожнього слова *P* в його кінець.

Наприклад:

	a	b	c	b			→			b	c	b	a	

Рішення.

Для вирішення цієї задачі пропонується виконати наступні дії:

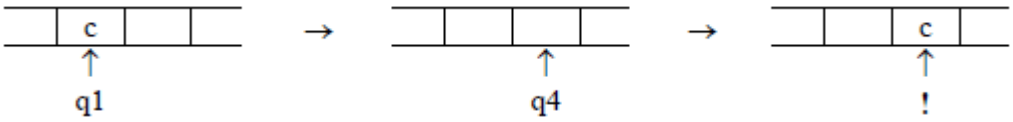
1. Запам`ятати перший символ слова *P*, а потім стерти цей символ.
2. Перегнати автомат вправо під першу порожню клітинку за *P* і записати в неї символ, що запам`ятали.

Як «бігти» вправо, ми вже знаємо з попереднього прикладу. А ось як запам`ятати перший символ? В МТ немає іншого запам`ятовуючого пристрою, окрім стрічки, а запам`ятовувати символ в якійсь клітинці на стрічці немає сенсу: як тільки автомат зсунеться вліво або вправо від цієї клітинки, він в той же час забуде даний символ. Що робити? Вихід тут такий – треба використовувати різні стани автомату. Якщо перший символ – це *a*, то треба перейти в стан *q2*, в якому автомат біжить вправо і записує в кінці *a*. Якщо ж першим був символ *b*, тодіпотрібно перейти в стан *q3*, де робиться все те ж саме, тільки в кінці записується символ *b*. Якщо ж першим був символ *c*, тоді переходимо в стан *q4*, в якому автомат дописує за вхідним словом символ *c*. Отже,те, яким був перший символ, ми фіксуємо переводом автомату в різні стани. Це типовий прийом при програмуванні на МТ. З урахуванням сказаного програма буде такою:

	a	b	c	Λ	
q1	Λ,R,q2	Λ,R,q3	Λ,R,q4	,R,	анализ 1-го символа, удаление его, разветвление
q2	,R,	,R,	,R,	a, ,!	запись *a* справа
q3	,R,	,R,	,R,	b, ,!	запись *b* справа
q4	,R,	,R,	,R,	c, ,!	запись *c* справа

Розглянемо поведінку цієї програми на вхідних словах, що містять не більше одного символу. При порожньому слові, яке є «поганим» дляз адачі, програма зациклиться – автомат, знаходячись в стані *q1* і потрапляючи весь час на порожні клітинки, буде бескінечно пересуватися вправо. (В цьому випадку програму можна було б зупинити, але ми спеціально зробили зациклювання, щоб продемонструвати таку можливість.)

Якщо ж у вхідному слові рівно один символ, тоді автомат зітре цей символ, зсунеться на одну клітинку вправо і запише в неї даний символ:



Таким чином, слово з одного символу просто зсунеться на клітинку вправо. Це припустимо. Бо клітинки стрічки не нумеровані, тому місцезнаходження слова на стрічці ніяк не фіксується і переміщення слова вліво або вправо замітити не можна. В зв`язку з цим немає необхідності, щоб вихідне слово обов`язково знаходилося в тому ж місці, де було вхідне слово, результат може виявитися і лівіше, і правіше цього місця.

Приклад 3 (порівняння символів, стирання слова)

A={*a,b,c*}. Якщо перший і останній символи (непорожнього) слова *P* однакові, тоді це слово не змінювати, а інакше замінити його на порожнє слово.

Рішення.

Для вирішення цієї задачі пропонується ввиконати наступні дії:

1. Запам`ятати перший символ вхідного слова, не стираючи його.
2. Перемістити автомат під останній символ і порівняти його з тим, що запам`ятали. Якщо вони рівні, то більше нічого не робити.
3. В протилежному випадку стерти все вхідне слово.

Як запам`ятати символ і як переганяти автомат під останній символ слова,

Ми вже знаємо з попередніх прикладів. Стирання же вхідного слова реалізується заміною всіх його символів на символ Λ. При цьому, раз вже автомат виявився в кінці слова, будемо переміщувати автомат справа наліво до першої порожньої клітинки. Ця дія описується наступною програмою для МТ (нагадаємо, що хрестик в комірці таблиці означає неможливість появи відповідної конфігурації при виконанні програми):

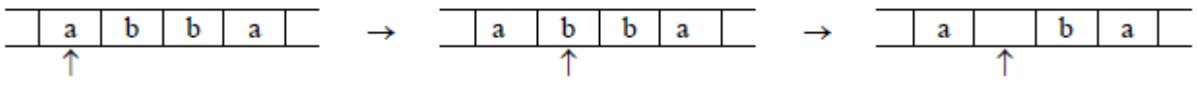
	a	b	c	Λ	
q1	, ,q2	, ,q4	, ,q6	, ,!	анализ 1-го символа, разветвление
q2	,R,	,R,	,R,	, L,q3	идти к последнему символу при 1-м символе *a*
q3	, ,!	, , q8	, , q8	×	сравнить посл. символ с *a*, не равны – на q8 (стереть *P*)
q4	,R,	,R,	,R,	, L,q5	аналогично при 1-м символе *b*
q5	, , q8	, ,!	, , q8	×	
q6	,R,	,R,	,R,	, L,q7	аналогично при 1-м символе *c*
q7	, , q8	, , q8	, ,!	×	
q8	Λ,L,	Λ,L,	Λ,L,	, ,!	стереть всё слово, двигаясь справа налево

Приклад 4 (видалення символу зі слова)

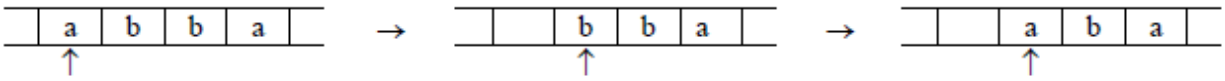
A={*a,b*}. Видалити зі слова *P* його другий символ, якщо такий є.

Рішення.

Здавалося б, цю задачу вирішити просто: треба зсунути автомат під клітинку з другим символом і потім очистити цю клітинку:



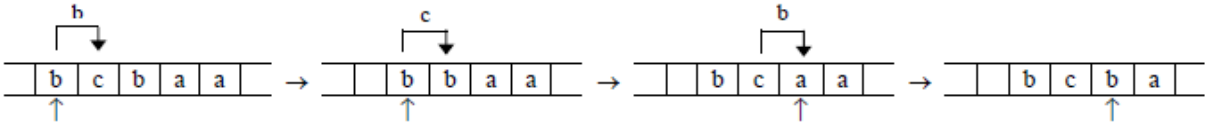
Однак нагадаємо, що всередині вихідного слова не повинно бути порожніх клітинок. Тому після видалення другого символу потрібно «стягнути» слово, переносячи перший символ на одну клітинку вправо. Для цього автомат повинен повернутися до першого символу, запам`ятати його і стерети, а потім, знову зсунувшись вправо, записати його в клітинку, де був другий символ. Однак початковий «похід» вправо до другого символу, щоб його стерти, і подальше повернення до першого символу є зайвими діями: яка різниця – переносити перший символ в порожню клітинку або в клітинку зякимось символом? Тому одразу запам`ятовуємо перший символ, стираємо його і записуємо замість другого символу:



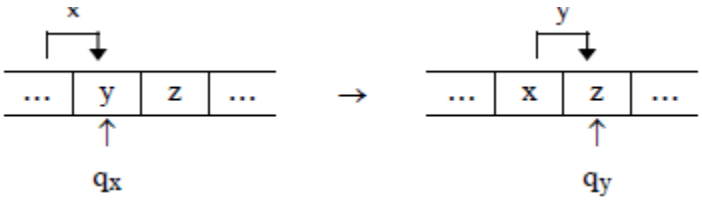
У вигляді програми для МТ все це записується так:

	a	b	Λ	
q1	Λ,R,q2	Λ,R,q3	, ,!	анализ и удаление 1-го символа, разветвление
q2	, , !	a, ,!	a, ,!	замена 2-го символа на *a*
q3	b, ,!	, , !	b, ,!	замена 2-го символа на *b*

Приклад 5 (стискання слова)
A={*a*,*b*,*c*}. Видалити зі слова *P* перше входження символу *a*, якщо таке є.
Рішення.
В попередньому прикладі ми переносили на позицію вправо тільки один символ. В даному ж прикладі ми будемо в циклі переносити вправо всі початкові символи *b i c* вхідного слова – до першого символа *a* або до порожньої клітинки:



Центральний момент тут – як перенести символ *x* з лівої клітинки в чергову клітинку, де знаходиться деякий символ *y*, щоб потім цей символ *y* можна було перенести в праву клітинку? Якщо через *qx* позначит стан, в якому в видиму клітинку треба записати символ *x*, що знаходився раніше в клітинці зліва, тоді цю дію можна зобразити так:



Для цього пропонується виконати такт *x,R,qy* , в якому поєднані наступні три дії: по-перше, в видиму клітинку записується символ *x*, що береться з клітинки зліва; по-друге, автомат зсувається вправо – під клітинку, в яку потім треба буде записати тількино замінений символ *y*; в-третіх, автомат переходить в стан *qy*, в якому він і буде виконувати цей запис. Повторення таких тактів в циклі і призведе до зсуву вправо на одну позицію початкових символів вхідного слова. Цей цикл повинен закінчитися, коли в черговій клітинці виявиться символ *a* абоΛ (*y*=*a* або*y*=Λ), а на початку циклу можна вважати, що на місце першого символу зліва переноситься символ «порожньо» (*x*=Λ). В кінці кінців отримаємо наступну програму для МТ:

	a	b	c	Λ	
q1	Λ,R,!	Λ,R,q2	Λ,R,q3	, ,!	$q_Λ$: стереть 1-й символ и перенести его вправо
q2	b, ,!	,R,	b,R,q3	b, ,!	q_b: запись *b*, перенос ранее видимого символа вправо
q3	c, ,!	c,R,q2	,R,	c, ,!	q_c: запись *c*, перенос ранее видимого символа вправо

В цій програмі слід звернути увагу на такт Λ,*R*,! , який виконується в конфігурації <*a*, *q1*>, тобто коли першим символом вхідногослова є *a*. Ясно, що треба просто стерти цей символ і зупинитися.

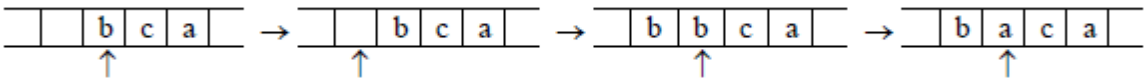
Однак в цьому такті вказаний ще зсув вправо. Навіщо? Нагадаємо, що в момент останову автомат повинен знаходитися під вихідним словом (під будь-яким його символом), тому ми і зсуваємо автомат з порожньої клітинки на клітинку з першим символом вихідного слова, який у вхідному слові був другим.

Приклад 6 (вставка символу в слово)

A={*a*,*b*,*c*}. Якщо*P* – непорожнє слово, то за його першим символом вставити символ *a*.

Рішення.

Ясно, що між першим і другим символами слова *P* потрібно звільнити клітинку для нового символу *a*. Для цього портібно перенести на одну позицію вліво перший символ (на старому місці його можна поки не видаляти), а потім, повернувшись на старе місце, записати символ *a*:



Перенос символу на одну позицію вліво аналогічний до переносу символу вправо, про що говорилося в двох попередніх прикладах, тому приведемо програму для МТ без додаткових коментарів. Зазначимо лише, що вставши на х *q2*, *q3* і *q4* автомат може бачити тільки порожню клітинку, а в стані *q5* він обов`язково бачить перший символ вхідного слова, але не порожню клітинку.

	a	b	c	Λ	
q1	,L,q2	,L,q3	,L,q4	, ,!	анализ 1-го символа для переноса его влево
q2	×	×	×	a,R,q5	приписать *a* слева
q3	×	×	×	b,R,q5	приписать *b* слева
q4	×	×	×	c,R,q5	приписать *c* слева
q5	, ,!	a, ,!	a, ,!	×	заменить бывший 1-й символ на *a*

Приклад 7 (розсування слова)

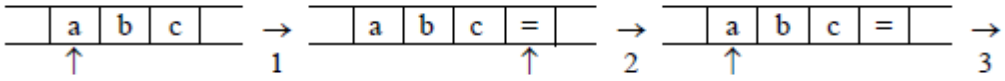
A={*a*,*b*,*c*}. Вставити в слово *P* символ *a* за першим входженням символу *c*, якщо таке є.

Рішення.

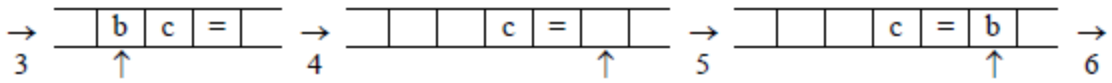
Продивляємося вхідне слово зліва направо до порожньої клітинки або до першого символу *c*. В першому випадку *c* не входить в *P*, тому нічого не робимо. В другому потрібно звільнити місце для вставляємого символу *a*, для чого зсуваємо початок слова *P* (від першого символу до знайденого символу *c*) на одну позицію вліво. При цьому виконуємо такий зсув справа наліво – від символу *c* до початку слова, раз вже автомат знаходиться під цим символом. Окрім тиого, щоб потім не повертатися до звільненої позиції, починаємо цей зсув з запису *a* замість знайденого символу *c*. Оскільки цей циклічний зсув вліво реалізується аналогічно циклічному зсуву вправо з прикладу 5, то не будемо пояснювати його, а одразу приведемо програму для МТ:

	a	b	c	Λ	
q1	,R,	,R,	a,L,q4	,L,!	вправо до *c*, вставка *a* вместо *c*, перенос *c* влево
q2	,L,	a,L,q3	a,L,q4	a, ,!	перенос *a* справа
q3	b,L,q2	,L,	b,L,q4	b, ,!	перенос *b* справа
q4	c,L,q2	c,L,q3	,L,	c, ,!	перенос *c* справа

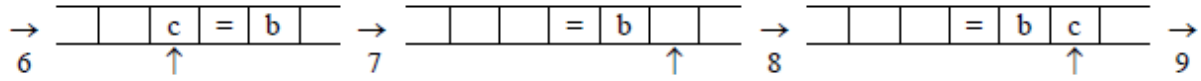
Приклад 8 (формування слова на новому місці)
A={*a*,*b*,*c*}. Видалити з *P* всі входження символу *a*.
Рішення.
Попередні приклади показують, що в МТ достатньо складно реалізуються вставки символів в слова і видалення символів зі слів. Тому іноді простіше не розсувати або стягувати вхідне слово, а формувати вихідне слово в вільному місці стрічки. Саме так ми і вчинимо при вирішенні даної задачі.
Конкретно пропонуються наступні дії:
1. Вихідне слово будемо будувати справа від вхідного. Щоб розмежувати ці слова, відділимо їх допоміжним символом, наприклад знаком =, відмінним від всіх символів алфавіту *A* (див. крок 1). (Нагадаємо, що на стрічці можуть бути записані не тільки символи з алфавіту вхідного слова.)
2. Після цього повертаємось до початку вхідного слова (див. крок 2).



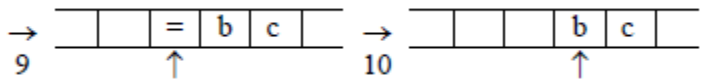
3. Тепер наша задача – перенести в циклі всі символи вхідного слова, окрім *a*, вправо за знак = в вихідне слово, що формується.
Для цього аналізуємо перший символ вхідного слова. Якщо це *a*, тоді стираємо його і переходимо до наступного символу (див. крок 3). Якщо ж перший символ – це *b* або *c*, тоді стираємо його і «біжимо» вправо до першої порожньої клітинки (див. крок 4), куди і записуємо цей символ (див. крок 5).



Знову повертаємося наліво до того символу, який став першим у вхідному слові, і повторюємо ті ж самі дії, але вже по відношенню до цього символу (див. кроки 6-9).



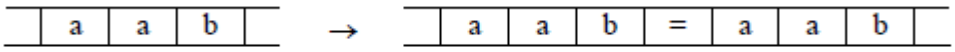
4. Цей цикл завершується, коли при поверненні наліво ми побачимо в якості першого символу знак =. Це ознака того, що ми неповністю продивилися вхідне слово і перенесли всі його символи, відмінні від *a*, і формуємо справа вихідне слово. Потрібно цей знак стерти, зсунутися вправо під вихідне слово і зупинитися (див. крок 10).



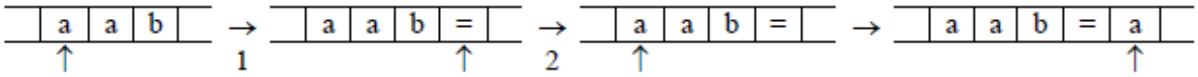
З урахуванням всього сказаного і будуємо програму для МТ. При цьому зазначимо, що окрім символів *a*, *b* і *c* в процесі вирішення задачі на стрічці з`являється знак =, тому в таблиці повинен бути передбачений стовпчик і для цього знаку.

	a	b	c	=	Λ	
q1	,R,	,R,	,R,	×	=, ,q2	записать справа знак =
q2	,L,	,L,	,L,	,L,	,R,q3	влево к 1-му символу слова
q3	Λ ,R,	Λ,R,q4	Λ,R,q5	Λ,R,!	×	анализ и удаление его, разветвление
q4	,R,	,R,	,R,	,R,	b, ,q2	запись *b* справа, возврат налево (в цикл)
q5	,R,	,R,	,R,	,R,	c, ,q2	запись *c* справа, возврат налево (в цикл)

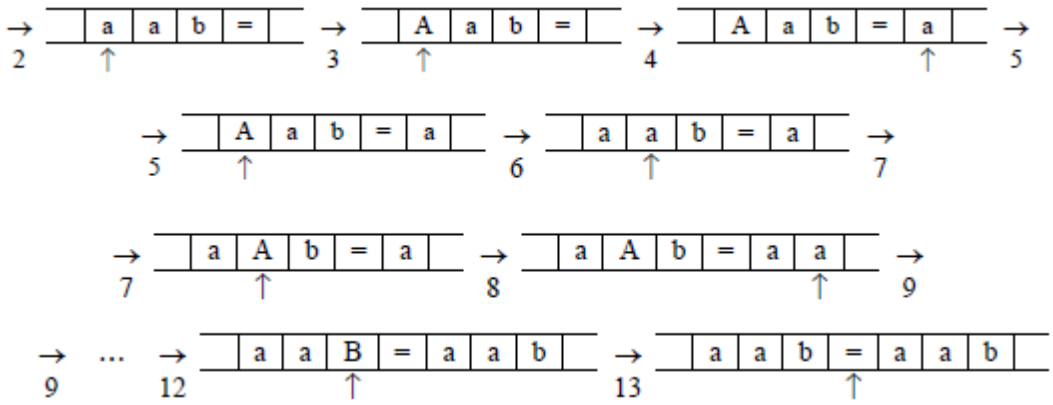
Приклад 9 (фиксування місця на стрічці)
$A=\{a,b\}$. Подвоїти слово P, поставивши між ним і його копією знак =.
Наприклад



Рішення.
Ця задача вирішується аналогічно попередній: в кінець вхідного слова записуємо знак =, потім повертаємося до початку слова і в циклі всі його символи (в тому числ і *a*) копіюємо в порожні клітинки справа:



Однак є і відмінність: символи вхідного слова, що копіюються, не видаляються, і це призводить до наступної проблеми. Записавши справа копію чергового символу, ми потім повинні повернутися до вхідного слова в позицію цього символу і потім зсунутися вправо до наступного символу, щоб скопіювати вже його. Але як дізнатися, в яку позицію вхідного слова потрібно повернутися? Наприклад, звідки ми знаємо в нашому прикладі, що після копіювання першого символу *a* ми повинні повернутися саме до першого символу вхідного слова, а не до другого або третього? В попередній задачі ми завжди поверталися до першого з залишившихся символів вхідного слова, а тепер ми зберігаємо всі символи, тому незрозуміло, які символи ми вже скопіювали, а які ще ні. Зазначимо також, що в МТ комірки стрічки ніяк не нумеруються, немає в МТ і лічильників, які б дозволили визначити, скільки символів ми вже скопіювали.
В загальному вигляді проблема, з якою ми стикнулися, наступна: як зафіксувати на стрічці деяку позицію, в якій ми вже були і до якої пізніше повинні повернутися? Зазвичай ця проблема вирішується так. Коли ми потрапляємо в цю позицію в перший раз, то замінюємо символ, що знаходиться ній, на його двійник – на новий допоміжний символ, причому різні символи замінюємо на різні двійники, наприклад *a* на *A* і *b* на *B*. Після цього ми виконуємо якісь дії в інших місцях стрічки. Щоби потім повернутися до нашої позиції, потрібно просто відшукати в стрічці ту клітинку, де знаходиться символ *A* або*B*. Потім в даній клітинці можна встановити попередній символ, якщо нам більше не потрібно фіксувати цю позицію (саме для відновлення попереднього символу і потрібно було замінювати різні символи на різні двійники).
Скористаємося цим прийомом в нашій задачі, виконуючи наступні дії:
1. Як вже сказано, спочатку записуємо знак = за вхідним словом (див. крок 1 вище).
2. Потім повертаємося під перший символ вхідного слова (див. крок 2 вище).
3. Далі замінюємо видимий символ *a* на двійник *A* (див.крок 3 нижче), «біжимо» вправо до першої вільної клітинки і записуємо в неї символ *a* (див.крок 4). Після цього повертаємося вліво до клітинки з двійником *A* (див. крок 5), відновлюємо попередній символ *a* і зсуваємось вправо до наступного символу (див. крок 6).



Тепер аналогічним чином копіюємо другий символ (замінюємо його на *A*,в кінець дописуємо *a* і т.д.) і всі наступні символи вхідного слова.

4. Коли ми скопіюємо останній символ вхідного слова і повернемося до його двійника (після кроку 12), то потім після зсуву на одну позицію вправо ми потрапимо на знак = (крок 13). Це сигнал про те, що вхідне слово повністю скопійоване, тому работу МТ потрібно завершити. З урахуванням всього сказаного отримуємо наступну програму для МТ:

	a	b	=	A	B	Λ	
q1	,R,	,R,	×	×	×	=,L,q2	поставить = справа от слова
q2	,L,	,L,	×	×	×	,R,q3	налево под 1-й символ
q3	A,R,q4	B,R,q5	, ,!	×	×	×	анализ и замена очередного символа
q4	,R,	,R,	,R,	×	×	a, ,q6	запись *a* справа
q5	,R,	,R,	,R,	×	×	b, ,q6	запись *b* справа
q6	,L,	,L,	,L,	a,R,q3	b,R,q3	×	возврат, восстановление, к след. символу

Зауважимо, що в цій програмі можна позбавитися від стану *q6*, якщо поєднати його зі станом *q2*, передбачивши в *q2* повернення вліво як до порожньої клітинки, так і до символів *A* і *B*:

	a	b	=	A	B	Λ	
			...				
q2	,L,	,L,	,L,	a,R,q3	b,R,q3	,R,q3	налево до Λ, A или B
			...				

Нормальні алгоритми Маркова

Короткий опис нормальних алгоритмів Маркова

Підстановки

Цікавою особливістю нормальних алгоритмов Маркова (НАМ) є те, що в них використовується лише одна елементарна дія – так звана підстановка, яка визначається наступним чином.

Формулою підстановки називається запис вигляду α→β (читається « α замінити на β»), де α і β – будь-які слова (можливо, і порожні). При цьому α називається лівою частиною формули, а β – правою частиною.

Сама ***підстановка*** (як дія) задається формулою поідстановки і застосовується до деякого слова *P*. Сенс операції зводиться до того, що в слові *P* відшукується частина, співпадаюча з лівою частиною цієї формули (тобтоз α), і вона замінюється на праву частину формули (тобто на β). При цьому інші частини слова *P*(зліва і справа від α) не змінюються. Отримане слово *R* називають ***результатом підстановки***. Умовно це можна зобразити так:

Необхідні уточнення:
1. Якщо ліва частина формули підстановки входить в слово *P*, то говорять, що ця формула ***застосовувана до P***. Але якщо α не входить в *P*, то формула вважається ***незастосовуваною до P***, і підстановка не виконується.
2. Якщо ліва частина α входить в *P* декілька разів, то на праву частину β, за визначенням, замінюється тільки перше входження α в *P*:
3. Якщо права частина формули підстановки – порожнє слово, то підстановкаα→ зводиться до викреслювання частини α з *P* (зазначимо попутно, що в формулах підстановки не прийнято як-небудь позначати порожнє слово):
4. Якщо в лівій частині формули підстановки вказано порожнє слово, то підстановка →β зводиться, за означенням, до дописування β зліва до слова *P*:

З цього правила випливає дуже важливий факт: формула з порожньою лівою частиною застосовувана до будь-якого слова. Зазначимо також, що формула з порожніми лівою і правою частинами не змінює слово.

Визначення НАМ
Нормальним алгоритмом Маркова (НАМ) називається непорожній кінцевий впорядкований набір формул підстановки:

В цих формулах можуть використовуватися два види стрілок: звичайна стрілка (→) і стрілка «з хвостиком» (а). Формула зі звичайною стрілкою називається звичайною ***формулою***, а формула зі стрілкою «з хвостиком» – ***заключною формулою***. Різниця між ними пояснюється нижче.
Записати алгоритм у вигляді НАМ – значить пред`явити такий набір формул.
Правила виконання НАМ
По-перше, задається деяке ***вхідне слово*** *P*. Д саме воно записане – не важливо, в НАМ це питання не оговорюється. Робота НАМ зводиться до виконання послідовності кроків. На кожному кроці формули, що входять до НАМ, продивляються зверху вниз і вибирається перша з формул, застосовуванихдо вхідного слова *P*, тобто сама верхня з тих, ліва частина яких входить в *P*. Далі виконується підстановка згідно зі знайденою формулою. Оримаємо нове слово *P′*. На наступному кроці це слово *P′* береться за вхідне і до нього застосовується та ж сама процедура, тобто формули знову проглядаються зверху вниз починаючи з самої верхньої і шукається перша формула, застосовувана до слова *P′*, після чого виконується ідповідна підстановка і отримуємо новее слово*P′′*. І так далі:

Слід звернути особливу увагу на той факт, що на кожному кроці формули в НАМ завжди проглядаються починаючи з самої першої.
Необхідні уточнення:
1. Якщо на черговому кроці була застосована звичайна формула (α→β), торобота НАМ продовжується.
2. Якщо ж на черговому кроці була застосована заключна формула(α а β), то після її застосування робота НАМ зупиняється. Те слово, яке вийшло в цей момент, і є ***вихідним словом***, тобто результат застосування НАМ до вхідного слова.
Як видно, різниця між звичайною і заключною формулами підстановки проявляється лише в тому, що після застосування звичайної формули робота НАМ продовжується, а після заключної формули – закінчується.
3. Якщо на черговому кроці до поточного слова не застосовувана жодна формула, то і в цьому випадку робота НАМ закінчується, а вихідним словом вважається поточне слово. Таким чином, НАМ зупиняється за двома причинами: або була застосована заключна формула, або жодна з формул не поідійшла. І те і інше вважається «хорошим»

завершенням роботи НАМ. В обох випадках говорять, що НАМ ***застосовуваний*** до вхідного слова. Однак може трапитися і так, що НАМ ніколи не зупиниться; це відбувається, коли на кожному кроці є застосовувана формула і ця формула звичайна. Тоді говорять, що НАМ ***незастосовуваний*** до вхідного слова. В цьому випадку про жодний результат не йдеться.

Приклади на складання НАМ

Розглянемо приклади, в яких демонструються типові прийоми складання НАМ.

Як і у випадку машини Тьюринга, для скорочення формулювання задач будемо використовувати наступні домовленості:

– літероюP будемо позначати вхідне слово;

– літерою *A* будемо позначати алфавіт вхідного слова, тобто набір тих символів, які і тільки які можуть входити у вхідне слово *P* (а в процесі виконання НАМ в словах, що обробляються, можуть з`являтися і інші символи). Крім того, в прикладах будемо справа від формул підстановки вказувати їх номери. Ці номери не входять до формул, а потрібні для посилань на формули при показі покрокового виконання НАМ.

Приклад 1 (вставка і видалення символів)

A={*a,b,c,d*}. В слові *P* треба замінити перше входження підслова *bb* на *ddd*

І видалити всі входження символу *c*.

Наприклад: abbcabbca → adddabba

Рішення.

По-перше зазначимо, що в НАМ, на відміну від машини Тьюринга, легко реалізуються вставки і видалення символів. Вставка нових символів в слово – це заміна деякого підслова на підслово з більшим числом символів; наприклад, за допомогою формули *bb→ddd* два символи будуть замінені на три символи. При цьому не треба турбуватися про те, щоб попередньо звільнити місце для додаткових символів, в НАМ слово разсувається автоматично. Видалення же символів – це заміна деякого підслова на підслово з меншим числом символів; наприклад, видаленн символу *c* реалізується

Формулою *c*→ (з порожньою правою частиною). При цьому ніяких порожніх позицій всередині слова не з`являється, стягування слова в НАМ відбувається автоматично.

$$\begin{cases} c \to & (1) \\ bb \to .ddd & (2) \end{cases}$$

Перевіримо алгоритм на словах:

$$\text{abb}\underline{c}\text{abbca} \xrightarrow{1} \text{abbabb}\underline{c}\text{a} \xrightarrow{1} \text{ab}\underline{bb}\text{abba} \xrightarrow{2} \text{adddabba}$$

Слово, яке вийшло після застосування завершальної формули (2), є вихідним словом, тобто результатом застосування НАМ до заданого вхідного слова.

Перевіримо наш НАМ ще і на вхідному слові, в яке не входить bb:

$$\text{d}\underline{c}\text{acb} \xrightarrow{1} \text{da}\underline{c}\text{b} \xrightarrow{1} \text{dab}$$

До останнього слова *(dab)* непридатна жодна формула, тому, згідно з визначенням НАМ, алгоритм зупиняється і це слово оголошується вихідним.

Приклад 2 (перестановка символів)

A={a, b}. Перетворити слово P так, щоб на його початку опинилися усі символи *a*, а у кінці - усі символи b.

Наприклад: babba → aabbb

Рішення.

Здавалося б, для вирішення цієї задачі потрібний складний НАМ. Проте це не так, завдання вирішується з допомогою НАМ, що містить всього одну формулу:

$$\{ ba \to ab$$

Поки в слові P справа хоч би від одного символу b є символ *a*, ця формула переноситиме *a* наліво від цього b. Формула перестає працювати, коли праворуч від b немає жодного *a*, це і означає, що усі *a* виявилися зліва від b. Наприклад:

babba → abbba → abbab → ababb → aabbb

Алгоритм зупинився на останньому слові, оскільки до нього вже непридатна наша формула.

Приклад 3 (використання спецзнаку)

A={*a, b*}. Видалити з непорожнього слова P його перший символ. Порожнє слово не змінювати.

Рішення.

$$\begin{cases} *a \mapsto & (1) \\ *b \mapsto & (2) \\ * \mapsto & (3) \\ \to * & (4) \end{cases}$$

Перевіримо роботу на словах: порожнє слово і *ававвa*.

1) порожнє слово:

$$\square \xrightarrow{4} \boxed{*} \xrightarrow{3} \square$$

2)

$$\textit{ававвa} \xrightarrow{4} \textit{*ававвa} \xrightarrow{1} \textit{ваввa}$$

Приклад 4 (фіксація спецзнаком замінюваного символу)

A={0,1,2,3}. Нехай P - непорожнє слово. Трактуючи його як запис невідємного цілого числа в чотирирічній системі числення, вимагається отримати запис цього ж числа, але в двійковій системі.

Наприклад: 0123 → 00011011

Рішення.

Як відомо, для переводу числа з чотирирічної системи в двійкову потрібно кожну цифру замінити на пару відповідних їй двійкових цифр: 0 → 00, 1 → 01, 2 → 10, 3 → 11.

Разом, отримуємо наступний алгоритм переводу чисел з чотирирічної системи в

двійкову:

$$\begin{cases} *0 \to 00* & (1) \\ *1 \to 01* & (2) \\ *2 \to 10* & (3) \\ *3 \to 11* & (4) \\ * \mapsto & (5) \\ \to * & (6) \end{cases}$$

Перевіримо цей НАМ на вхідному слові 0123:

$$0123 \xrightarrow{6} \underline{*}0123 \xrightarrow{1} 00\underline{*}123 \xrightarrow{2} 0001\underline{*}23 \xrightarrow{3} 000110\underline{*}3 \xrightarrow{4} 00011011\underline{*} \xrightarrow{5} 00011011$$

Перевірити самостійно на інших 2-х словах.

Приклад 5 (переміщення спецзнаку)
A={*a, b*}. Вимагається приписати символ *a* до кінця слова P.
Наприклад: bbab → bbaba
Рішення.

Передусім нагадаємо, що формула →*a* приписує символ *a* ліворуч до слова P, а не справа. Щоб приписати *a* справа, потрібно спочатку помітити кінець слова. Для цього скористаємося спецзнаком, який помістимо в кінець P, а потім замінимо його на *a*:

$$P \to \ldots \to P* \mapsto Pa$$

З урахуванням усього сказаного отримуємо наступний НАМ:

$$\begin{array}{ll} (1) & \\ (2) & \\ (3) & \\ (4) & \end{cases} \begin{array}{l} *a \to a* \\ *b \to b* \\ * \mapsto a \\ \to * \end{array}$$

Відмітимо, що при порожньому вхідному слові цей НАМ спочатку введе зірочку, а потім тут же замінить її на символ *a* і зупиниться.
Перевірити роботу на 2-хсловах.

Приклад 6 (зміна спецзнаку)
A={*a, b*}. У слові P замінить на *aa* останнє входження символу *a*, якщо такий є.
Наприклад: bababb → babaabb
Рішення.

Введемо спецзнаки # і * розподіливши між цими спецзнаками обов'язки: нехай * рухається управо, а # - вліво. З'явитися ж спецзнак # повинен тоді, коли * дійде до кінця слова, тобто коли праворуч від * не виявиться інших символів. Така заміна спецзнаку "виключить з гри" усі формули із зірочкою в лівій частині, тому вони вже не

заважатимуть формулам із спецзнаком #. Якщо так і зробити, то отримаємо наступний НАМ:

$$\begin{cases} *a \to a* & (1) \\ *b \to b* & (2) \\ * \to \# & (3) \\ b\# \to \#b & (4) \\ a\# \mapsto aa & (5) \\ \# \mapsto & (6) \\ \to * & (7) \end{cases}$$

Перевіримо цей алгоритм на вхідному слові *bababb*(подвійна стрілка означає декілька кроків застосування формул (1) і (2)):

$$\underset{}{bababb} \xrightarrow{7} *bababb \overset{1,2}{\Rightarrow} bababb* \xrightarrow{3} bababb\# \xrightarrow{4} babab\#b \xrightarrow{4} baba\#bb \xrightarrow{5} babaabb$$

Якщо ж у вхідне слово не входить символ *a*, тоді маємо:

$$bb \xrightarrow{7} *bb \xrightarrow{2} b*b \xrightarrow{2} bb* \xrightarrow{3} bb\# \xrightarrow{4} b\#b \xrightarrow{4} \#bb \xrightarrow{6} bb$$

Приклад 7 (перенесення символу через слово)

A={a, b}. Перенести в кінець непорожнього слова P його перший символ. Порожнє слово не міняти.

Наприклад: bbaba → babab

Рішення.

Усі ці дії реалізуються у вигляді наступного НАМ:

$$\begin{cases} *a \to A & (1) \\ *b \to B & (2) \\ Aa \to aA & (3) \\ Ab \to bA & (4) \\ Ba \to aB & (5) \\ Bb \to bB & (6) \\ A \mapsto a & (7) \\ B \mapsto b & (8) \\ * \mapsto & (9) \\ \to * & (10) \end{cases}$$

Тут формули (1) і (2) замінюють перший символ слова (разом з *) на A або B. Формули (3) -(6) переганяють A і B в кінець слова. Формули (7) і (8) застосовуються тільки тоді, коли A і B виявляться у кінці слова (коли за ними вже немає символів), і відновлюють початковий вид першого символу. Формула (9) потрібна на випадок порожнього вхідного слова. Формула ж (10) ставить спецзнак * перед першим символом.

Перевіримо цей алгоритм на вхідному слові bbaba і на вхідному слові з одного символу:

$$bbaba \overset{10}{\rightarrow} \underline{*b}baba \overset{2}{\rightarrow} \underline{Bb}aba \overset{6}{\rightarrow} b\underline{Ba}ba \overset{5}{\rightarrow} ba\underline{Bb}a \overset{6}{\rightarrow} bab\underline{Ba} \overset{5}{\rightarrow} baba\underline{B} \overset{8}{\mapsto} babab$$

$$a \overset{10}{\rightarrow} \underline{*a} \overset{1}{\rightarrow} \underline{A} \overset{7}{\mapsto} a$$

Інше рішення

Відмітимо, що в цьому НАМ можна зменшити число формул, якщо не вводити нові символи A і B, а використовувати замість них пари **a* і *b. Це дозволить виключити з НАМ формули (1) і (2), які вводять символи A і B, і формули (7) і (8), які відновлюють перший символ. Що ж до "перестрибування" цих пар через якийсь символ то воно реалізується формулами виду *aξ→ξ*a і *bξ→ξ*b. При цьому ніякої плутанини між першим символом слова і іншими символами не буде, оскільки тільки перед першим символом знаходиться зірочка.

Отже, можливий наступний НАМ, вирішує нашу задачу:

$$\begin{cases} *aa \rightarrow a*a & (1) \\ *ab \rightarrow b*a & (2) \\ *ba \rightarrow a*b & (3) \\ *bb \rightarrow b*b & (4) \\ * \mapsto & (5) \\ \rightarrow * & (6) \end{cases}$$

Перевіримо цей алгоритм на томуж вхідному слові bbaba :

$$bbaba \overset{6}{\rightarrow} \underline{*bb}aba \overset{4}{\rightarrow} b\underline{*ba}ba \overset{3}{\rightarrow} ba\underline{*bb}a \overset{4}{\rightarrow} bab\underline{*ba} \overset{3}{\rightarrow} baba\underline{*}b \overset{5}{\mapsto} babab$$

Приклад 8 (використання декількох спецзнаків)

A={a, b}. Подвоїти слово P, тобто приписати до P (ліворуч або справа) його копію. Наприклад: abb → abbabb

Рішення.

НАМ:

$$
\begin{cases}
a \to a & (1)\\
b \to b & (2)\\
* \to = & (3)\\
Aa \to aA & (4)\\
Ab \to bA & (5)\\
A= \to =A & (6)\\
A \to a & (7)\\
Ba \to aB & (8)\\
Bb \to bB & (9)\\
B= \to =B & (10)\\
B \to b & (11)\\
\#a \to a\#A & (12)\\
\#b \to b\#B & (13)\\
\#= \mapsto & (14)\\
\to \#* & (15)
\end{cases}
$$

Тут формули (1) -(3) "переганяють" зірочку в кінець вхідного слова і замінюють її на символ =. Формул (4) -(7) переганяють символ А в кінець слова, після чого замінюють на *a*. Формули (8) -(11) роблять те ж саме з B і b. Формули (12 і (13) "вводять в гру" символи А і В, відповідні тому символу вхідного слова, який має бути скопійований наступним. Формула (14) застосовується тільки тоді, коли праворуч від # немає ні *a*, ні b, тобто коли повністю проглянуто усе вхідне слово. І, нарешті, формула (15) вводить відразу два спецзнаки # і *.

Перевіримо цей алгоритм на двох вхідних словах - на порожньому і на abb:

$$
\text{<пустое слово>} \xrightarrow{15} \#\underline{*} \xrightarrow{3} \underline{\#=} \xrightarrow{14} \quad \text{(т.е. получили <пустое слово><пустое слово>)}
$$

$$
\begin{aligned}
abb &\xrightarrow{15} \#*abb \xRightarrow{1,2} \#abb\underline{*} \xrightarrow{3} \underline{\#a}bb= \xrightarrow{12} a\#Abb= \xRightarrow{4\text{-}6} a\#bb=\underline{A} \xrightarrow{8} a\underline{\#b}b=a \xrightarrow{12}\\
&\xrightarrow{12} ab\#Bb= \xRightarrow{8\text{-}10} ab\#b=a\underline{B} \xrightarrow{11} ab\underline{\#b}=ab \xrightarrow{12} abb\#B=ab \xRightarrow{8\text{-}10} abb\#=ab\underline{B} \xrightarrow{11}\\
&\xrightarrow{14} abb\underline{\#=}abb \mapsto abbabb
\end{aligned}
$$

Інше рішення

Приведемо ще одне рішення задачі подвоєння слова, в якому пропонується виконати наступні дії. (Перевірити роботу на 2-х словах)

$$
\begin{cases}
a & \to aA \\
b & \to bB \\
* & \to \# \\
Aa & \to aA \\
Ab & \to bA \\
Ba & \to aB \\
Bb & \to bB \\
A\# & \to \#a \\
B\# & \to \#b \\
\# & \mapsto \\
 & \to *
\end{cases}
$$

Питання для опрацювання.

1. Скінченні автомати.
2. Автомати з машинною пам’яттю.
3. Машини Тьюрінга.
4. Автомати Маркова.
5. Машина Поста.

Варіанти завдань.

2.1. Скласти програму роботи для Машини Тьюринга, де P – вхідне непорожнє слово (length(P)>2), A – алфавіт вхідного слова, тобто набір символів з яких може складатися слово P (табл. 2.1).

Таблиця 2.1

№	Завдання
1	$A=\{0,1,2,3,4,5,6,7,8,9\}$. Збільшити задане на стрічці слово P на 1.
2	$A=\{a,b\}$. Видалити зі слова P його другий символ.
3	$A=\{a,b,c\}$. Видалити зі слова P перше входження символу b, якщо воно існує.
4	$A=\{a,b,c\}$. За першим символом слова P вставити символ c.
5	$A=\{a,b,c\}$. *Вставит*и в слово P символ a за першим входженням символу c, якщо воно існує.
6	$A=\{a,b,c\}$. Видалити зі слова P всі входження символу a.
7	$A=\{a,b\}$. Продублювати слово P, розділяючи слово і копію знаком =.
8	$A=\{a,b,c\}$. Дописати зліва до слова P символ b ($P \to bP$).
9	$A=\{a,b,c\}$. Дописати справа до слова P символи bc ($P \to Pbc$).
10	$A=\{a,b,c\}$. Замінити на c кожен другий символ в слові P.
11	$A=\{a,b,c\}$. Залишити в слові P тільки символи b.
12	$A=\{a,b,c\}$. Залишити в слові P тільки останній символ.
13	$A=\{0,1\}$. Вважати слово P записом двійкового числа. Видалити з нього незначущі нулі, якщо такі є.
14	$A=\{a,b,c\}$. Якщо P – слово парної довжини $2n$, то у відповіді вивести a, в протилежному випадку – порожнє слово.

15	*A={a,b,c}.* Дописати зліва до слова *P* його перший символ.
16	*A={a,b}. В слов*і *P* поміняти місцями його перший і останній символи.
17	*A={a,b}. Замінити в* слові *P* кожне входження *b* на *aa*.
18	*A={a,b,c}. Замінити в* слові *P* кожне входження *ab* на *c*.
19	*A*={*a*,*b*}. Подвоїти слово *P*. Наприклад: *abb* → *abbabb*.
20	*A={a,b}.* Подвоїти кожний символ в слові *P*. Наприклад: *bab* → *bbaabb*.
21	*A={0,1}. Перевернути слово P.* Наприклад: 011 → 110.
22	*A*={0,1}. Перенести перший символ слова *P* в кінець слова.
23	*A*={0,1}. Перенести останній символ слова *P* на початок слова.
24	*A*={0,1}. В слові *P* поміняти місцями перший і останній символи.
25	*A*={0,1}. Якщо в слові *P* співпадають перший і останній символи, то видалити обидва, в протилежному випадку слово не змінювати.
26	*A*={*a*,*b*,*c*}. Зі свіх входжень символу *a* в слово *P* залишити останнє входження, якщо таке існує.
27	*A*={*a*,*b*,*c*}. Якщо слово *P* починається з символу *b*, то замінити *P* на порожнє слово, в протилежному випадку слово не змінювати.
28	*A={0,1,2,3,4,5,6,7,8,9}.* Зменшити задане на стрічці слово *P* на 1.
29	*A*={0,1}. Замінити в слові *P* кожне входження 00 на 1.
30	*A*={0,1}. Залишити в слові *P* тільки символи 0.

2.2. Скласти нормальний алгоритм Маркова, де *P* – вхідне непорожнє слово (length(*P*)>2), *A* – алфавіт вхідного слова, тобто набір символів з яких може складатися *P* (табл. 2.2).

Таблиця 2.2

№	Завдання
1	*A={a,b,c,d}. В слов*і *P* замінити перше входження символів *aa* на *bbb* та видалити всі входження символу *c*.
2	*A={a,b}.* Перетворити слово *P* так, щоб на початку опинилися всі символи *a*, а в конці – всі символи *b*. Наприклад: babba → aabbb.
3	*A={a,b}.* Видалити зі слова *P* його перший символ.
4	*A*={0,1,2,3}. Слово *P* – невід'ємне ціле число в четвірковій системі зчислення. Необхідно записати це число в двійковій системі зчислення. Наприклад: 0123 → 00011011.
5	*A={a,b}.* Приписати символ *a* в кінець слова *P*. Наприклад: bbab → bbaba.
6	*A={a,b}. В слов*і *P* замінити останнє входження символу *a* на *aa*, якщо таке існує. Наприклад: bababb → babaabb.
7	*A={0,1}. Перенести в к*інець слова *P* його перший символ. Наприклад: 10001 → 00011.
8	*A={a,b}.* Продублювати слово *P*. Наприклад: abb → abbabb.
9	*A*={*m*,*n*,*k*}. В слові *P* замінити всі пари *nk* на *m*, якщо такі є.

10	*A*={*f*,*h*,*p*}. В слові *P* замінити на *m* тільки першу пару *nk*, якщо такі є. *A={a,b,c}.* Дописати слово *ccc* зліва до слова *P*.
12	*A*={0,1}. В слові *P* поміняти місцями перший і останній символи.
13	*A={a,b,c}. Зам*інити слово *P* на слово *a*.
14	*A={1,2,3*,4}. Перетворити слово *P* таким чином, щоб спочатку були всі парні цифри, а потім – всі непарні. Наприклад: 432214→422431.
15	*A*={*a*,*b*,*c*}. }. Перетворити слово *P* таким чином, щоб спочатку були всі символи *a*, потім – всі символи *b* і в кінці – всі символи *c*.
16	*A={a,b,c}.* В слові *P* подвоїти перший символ, тобто приписати цей символ зліва до *P*.
17	*A={a,b,c}.* Зі слова *P* видалити другий символ.
18	*A={a,b,c}. За пер*шим символом слова *P* включити символ *c*.
19	*A={a,b,c}.* Дописати слово *cba* справа до слова *P*.
20	*A={a,b,c}.* Видалити зі слова *P* його останній символ.
21	*A={a,b}. В слов*і *P* всі символи *a* замінити символами *b*, а всі початкові символи *b* – на *a*.
22	*A*={*a*,*b*,*c*}. Подвоїти кожний символ в слові *P*. Наприклад: *bacb* → *bbaaccbb*.
23	*A*={*a*,*b*,*c*}. Видалити зі слова *P* друге вхождення символу *a*, якщо такий існує.
24	*A*={*a*,*b*,*c*}. Видалити зі слова *P* третє вхождення символу *a*, якщо такий існує.
25	*A*={*a*,*b*}. Перенести перший символ слова *P* в кінець слова.
26	*A*={*a*,*b*}. Перенести останній символ слова *P* на початок слова.
27	*A*={0,1}. Перевернути слово *P*. Наприклад: 011 → 110.
28	*A*={*a*,*b*,*c*}. Зі свіх входжень символу *a* в слово *P* залишити останнє входження, якщо таке існує.
29	*A*={0,1}. Якщо в слові *P* співпадають перший і останній символи, то видалити обидва, в протилежному випадку слово не змінювати.
30	*A*={*a*,*b*,*c*}. Видалити зі слова *P* всі вхождення символу *a*, якщо такі існують.