কমপিউটার পাঠশালা

নিজে নিজে বেসিক শিখুন

(শেষ পর্ব)

মোঃ আসাদুর রহমান

বেসিক শেখার শেষ পর্যায়ে আজ আমি আপনাদের কিছু বেসিক কীওয়ার্ড-এর সাথে পরিচয় করিয়ে দেব IIF ... THEN, FOR ... NEXT, RENUM, READ ... DATA, LOAD এবং SAVE— এই কীওয়ার্ডগুলো কিছু উদাহরণের মাধ্যমে আমি এখানে আলোচনা করব। প্রয়োজনবোধে আপনারা পূর্ববর্তী সংখ্যাগুলোর সাহায্য নিতে পারেন।

```
10 LET X = 0
20 LET X = X + 1
30 PRINT X
40 GOTO 20
```

উপরের প্রোগ্রামটি RUN করলে আপনি 1, 2, 3 ... সংখ্যাগুলো ক্রমান্বয়ে দেখতে পাবেন। সংখ্যাগুলো একের পর এক আসতেই থাকবে যতক্ষণ না আপনি RUN/STOP (Ctrl Break) চাবিটি চাপছেন। একটি বেসিক স্টেটমেন্ট এই প্রোগ্রামে যোগ করে প্রোগ্রামে এমন শর্ত আরোপ করা যায় যা উপরোক্ত প্রক্রিয়াটিকে একটি নির্দিষ্ট সংখ্যা আসার পর থামিয়ে দেবে। ধরুন আপনি 1 থেকে 5 পর্যন্ত সংখ্যাগুলো দেখতে চাইছেন, সেক্ষেত্রে সহজ কথায় শর্তটি হবে— 'যখন X-এর মান 5 হবে, তখন প্রক্রিয়াটি থামাও'। উপরের প্রোগ্রামে আমরা দেখছি, লাইন 40 প্রোগ্রামের গতি প্রবাহকে লাইন 20 এর দিকে পরিচালন করছে। এ কারণেই আমরা 1, 2, 3, 4 ... এভাবে অন্তহীন সংখ্যা দেখতে পাচ্ছি। অতএব সবচেয়ে ভাল হবে যদি শর্তটি লাইন 30 এবং 40-এর মাঝে আরোপ করা যায়। এখন নীচের শর্তযুক্ত স্টেটমেন্টটি প্রোগ্রামে যোগ করে আপনি আবার প্রোগ্রামটি চালান।

```
35 IF X = 5 THEN STOP
```

এই নতুন স্টেটমেন্টটি যোগ করার পর সম্পূর্ণ প্রোগ্রামটি দেখাবে এরকম ঃ

```
Prog. 3A
10 LET X = 0
20 LET X = X + 1
30 PRINT X
35 IF X = 5 THEN STOP
40 GOTO 20
RUN
1
2
3
4
5
OK
```

IF স্টেটমেন্টটি সংযোজনের মাধ্যমে আপনি কমপিউটারকে দিলেন সিদ্ধান্ত গ্রহণ এবং তদনুযায়ী পদক্ষেপ নেওয়ার সত্যিকার ক্ষমতা।

IF স্টেটমেন্টের ফরম্যাট ঃ

IF Condition THEN action

Condition (শর্ত) যে কোন string, variable এবং expression-এর মধ্যে হতে পারে। Action হতে পারে যে কোন বেসিক কীওয়ার্ড (প্রয়োজনে সঙ্গে কোন আরগুমেন্টও থাকতে পারে)।

IF স্টেটমেন্টের কিছু উদাহরণ এখানে দেওয়া হল:

	শর্ত (Condition) সত্য হবে যখন...
IF X = 20 THEN ...	X-এর মান 20 হবে।
IF N+10 = 20 THEN...	N+10-এর মান 20-এর সমান হবে।
IF A=B THEN...	A-এর মান B-এর মানের সমান হবে।
IF Y+10 <20 THEN...	Y+10-এর মান 20-এর ছোট হবে।
IF TS="ALI" THEN...	TS এর ষ্ট্রীং এর মান হবে "ALI"
IF TS<AS+BS THEN...	TS এর ষ্ট্রীং AS এবং BS এর ষ্ট্রীং (দুইটি ষ্ট্রীং যোগ করে) এর চেয়ে ছোট হবে।
IF x <> 100 THEN ...	X এর মান 100 এর সমান নয়।
IF N>=80 THEN ...	N এর মান 80 এর চেয়ে বড় অথবা 80 এর সমান।

একই IF স্টেটমেন্টে আপনি একসাথে কয়েকটি শর্ত দিতে পারেন, যেমন —

IF A > B AND A > C	A এর মান B এবং C উভয়ের মানের চেয়ে বড়।
IF A > B OR A > C	A এর মান B ভেরিয়েবল এর চেয়ে বড় অথবা A এর মান C ভেরিয়েবলের চেয়ে বড়।
IF RATE NOT = 16.5	ভেরিয়েবল RATE এর মান 16.5 এর সমান নয়।

AND, OR ও NOT এগুলো বেসিক কীওয়ার্ড এবং লজিকাল অপারেটর (logical operator) নামে পরিচিত।

যখন একটি শর্ত সত্য প্রতিপন্ন হয়, তখন বেসিক THEN এর পরে লেখা কীওয়ার্ডগুলোর কাজ সম্পাদন করে। যদি শর্তটি সত্য না হয়, তাহলে THEN এর পরের action এর কাজ হয় না তার পরিবর্তে IF স্টেটমেন্টের পরের স্টেটমেন্টটি সম্পাদন করে।

আপনি 'THEN' এর পরে action হিসেবে যেকোন কীওয়ার্ড (স্টেটমেন্ট) ব্যবহার করতে পারেন। একই সাথে কয়েকটি স্টেটমেন্ট লিখতে পারেন এক্ষেত্রে প্রতিটি স্টেটমেন্ট ":" চিহ্নের মাধ্যমে আলাদা করতে হবে।

action হিসেবে কীওয়ার্ড ব্যবহারের কিছু উদাহরণ আমরা এখন দেখব:

	(শর্ত) condition সত্য হলে যে action নেওয়া হবে
THEN STOP	প্রোগ্রাম execution থেমে যাবে।
THEN PRINT "BYE BYE"	"BYE BYE" এই ষ্ট্রিংটি প্রিন্ট করবে।
THEN GOTO 20	লাইন 20 এ ব্রাঞ্চ (brunch) করবে।
THEN LET N=6	ভেরিয়েবল N এর মান হবে 6
THEN S=S+X	ভেরিয়েবল S এর মান হবে S ও X এর মানের যোগফল।
THEN LET AS="Good"	"Good" ষ্ট্রিংটি হবে ষ্ট্রিং ভেরিয়েবল AS এর মান
THEN H=X:MS="Try again"	X এর মান ভেরিয়েবল H এ এবং "Try again ষ্ট্রিংটি MS ভেরিয়েবলে ষ্টোর হবে।
THEN ?"VERY GOOD" : P=P+1	"VERY GOOD" ষ্ট্রিংটি প্রিন্ট করা হবে এবং P ভেরিয়েবল এর মান 1 বাড়ানো হবে।
THEN A=B:B=C:GOTO 150	A তে B এর এবং B তে C এর মান store করা হবে এবং লাইন ১৫০ এ ব্রাঞ্চ করবে।

এবার আপনি নীচের পরিবর্তনগুলো সংযুক্ত করে Prog. 3A প্রোগ্রামটি আবার লিখুন:

```
Prog. 3B
10 CLS
20 PRINT "NUMBERS FROM 1 TO 10"
30 LET X = 0
40 LET X = X + 1
50 PRINT X
60 IF X < > 10 THEN GOTO 40
70 END
```

উপরের প্রোগ্রামটি সবসময়ই 1 থেকে 10 পর্যন্ত সংখ্যাগুলো প্রিন্ট করবে। কিন্তু আপনি যদি ১ থেকে শুরু করে যেকোন সংখ্যা (ব্যবহারকারীর ইচ্ছানুযায়ী) পর্যন্ত প্রিন্ট করাতে চান, তাহলে আপনাকে প্রোগ্রামে INPUT স্টেটমেন্ট ব্যবহার করতে হবে INPUT স্টেটমেন্টের মাধ্যমে আপনি প্রোগ্রাম ব্যবহারকারীকে একটি সংখ্যা কমপিউটারে প্রবেশ করানোর সুযোগ দিতে পারবেন। আপনি উপরের প্রোগ্রামে একটি INPUT স্টেটমেন্ট যোগ করে ইউজার (প্রোগ্রাম ব্যবহারকারী) এর কাছ থেকে একটি সংখ্যা ইনপুট হিসেবে নিতে পারেন। সেই একই সংখ্যাকে IF স্টেটমেন্টে ব্যবহার করে আপনি প্রিন্ট করার প্রক্রিয়াটি ঐ বিশেষ সংখ্যায় থামানোর ব্যবস্থা করতে পারেন। প্রোগ্রামে এই পরিবর্তনের মাধ্যমে আপনার প্রোগ্রাম এখন ১ থেকে ইনপুট অনুযায়ী যেকোন সংখ্যা পর্যন্ত প্রিন্ট করতে সক্ষম হবে।

prog. 3B এর 20 নম্বর লাইনটি নতুন করে লিখুন, 20 PRINT "NUMBERS FROM 1 TO ANY NUMBER" এরপর নীচের লাইনটি প্রোগ্রামে সংযোগ করুন, 25 INPUT 'Enter a number to stop at" ; num এবার লাইন 60 কে বদলে দিন, 60 IF X < > num THEN GOTO 40 এখন প্রোগ্রামটি RUN করান।

```
RUN
NUMBERS FROM 1 TO ANY NUMBER
Enter a number to stop at ? 5
1
2
3
4
5
ok
```

নীচের প্রোগ্রামটি 2 থেকে 20 পর্যন্ত সবগুলো জোড় সংখ্যা প্রিন্ট করবে।

```
Prog. 4A
10 CLS
20 PRINT "EVEN NUMBERS FROM 2 TO 20"
30 X = X + 2
40 PRINT X
50 IF X < > 20 THEN GOTO 30
60 END
```

এবার আর একটি প্রোগ্রাম লিখছি যা 10 থেকে 1 পর্যন্ত সংখ্যাগুলো প্রিন্ট করবে।

```
Prog. 4B
10 CLS : X = 11
20 PRINT "NUMBERS FROM 10 TO 1"
30 X = X -1
40 PRINT X
50 IF X < > 1 THEN GOTO 30
60 END
```

অনেক ক্ষেত্রে একটি কার্য সম্পাদনের জন্য কতগুলো ইন্সট্রাকশন বারবার

execute করার প্রয়োজন হয় (যেমনProg. 4A ও Prog. 4B প্রোগ্রাম দুইটিতে 30 ও 40 নম্বর লাইনগুলো 10 বার execute করতে হয়) এই পুনরাবৃত্তির পদ্ধতিকে লুপিং (looping) বলা হয়।

```
        10 CLs                              10 CLS
->      20 PRINT "DHAKA-1205"       ->      20 X = X + 1
End :           :                           30PRINT "DHAKA-1205"
Less :          :                   Loop :              :
Loop :          :                   for  :              :
—>      80 GOTO 20                  5 times :           :
        90 END                      —>      80 IF x < > 5 THEN GOTO 20
                                            90 END
```

উপরের বাদিকের প্রোগ্রামটি অবিরত "DHAKA-1205" এই স্ট্রিংটি প্রিন্ট করতে থাকবে আর ডান দিকের প্রোগ্রামটি সেই একই কাজ করবে ৫ বার।

ডান দিকের প্রোগ্রামে ২০ নম্বর লাইনে হিসেব রাখা হচ্ছে কতবার প্রোগ্রামটি এই লাইনের স্টেটমেন্ট অতিক্রম করবে। যতবার প্রোগ্রামটি ঘুরে ঘুরে এই লাইনে আসবে, ততবার ভেরিয়েবল X এর মান ১ করে বৃদ্ধি পাবে। এই ধরণের ভেরিয়েবল কে বলা হয় কাউন্টার (counter)। লাইন 30 এর কাজ হচ্ছে "DHAKA-1205" স্ট্রিংটি প্রিন্ট করা। 80 নম্বর লাইনে IF স্টেটমেন্ট ব্যবহার করে শর্ত আরোপ করা হয়েছে, যাতে প্রোগ্রামটি বারবার লাইন 20 এ ফিরে আসবে যতক্ষণ পর্যন্ত X–এর মান 5 এর সমান না হচ্ছে। লুপের (LOOP) মধ্যস্থিত স্টেটমেন্টগুলোকে সাধারণতঃ স্টেটমেন্ট ব্লক (Block of statement) বলা হয়। X এর মান যখন 5 হয়, তখন লাইন 80 এর শর্ত (Condition) মিথ্যা প্রতিপন্ন হয় এবং প্রোগ্রামটি তখন লাইন 90 এর END স্টেটমেন্ট কার্যকর করতে চলে যায়। লক্ষ্য করুন prog 4A ও 4B এর 30 ও 40 নম্বর লাইনগুলো একটি লুপের মধ্যে পড়ে।

কাউন্টার ও শর্ত ব্যবহার ছাড়াও বেসিকে লুপ তৈরির আরও একটি উপায় আছে। তা হল FOR ও NEXT কীওয়ার্ডগুলো ব্যবহার করে। FOR ও NEXT স্টেটমেন্ট সবসময় একসাথে ব্যবহৃত হয় অর্থাৎ একটি FOR স্টেটমেন্টের জন্য অবশ্যই একটি NEXT স্টেটমেন্ট থাকতে হবে। একটি লুপের ভেতর আর একটি লুপ, তার ভেতর আর একটি লুপ এভাবে আপনি কয়েকটি লুপ তৈরি করতে পারেন। লুপের ভেতর লুপ এধরণের লুপকে nested (নেস্টেড) লুপ বলে। একটি লুপের ভেতর আপনি লুপ ভেরিয়েবল ব্যবহার করতে পারবেন, কিন্তু কোন অবস্থাতেই এর মান পরিবর্তন করবেন না। আর কখনই কোন FOR লুপের ভেতর থেকে বের হয়ে আসার চেষ্টা করবেন না FOR স্টেটমেন্ট শেষ করার আগে। FOR ও NEXT স্টেটমেন্ট ব্যবহার করে নীচে একটি লুপ দেখানো হল যা ১০ বার লুপের ভেতর ঘুরবে:

```
                    20 FOR X = 1 TO 10 STEP 1
    Loop —>         :           :
    for 10          :           :
t   imes            :           :
         —>         80 NEXT X
```

এখানে X হচ্ছে লুপ ভেরিয়েবল, লুপের প্রারম্ভে যার মান ১ লুপের শেষে এর মান (end value) ১০ এবং FOR স্টেটমেন্টের শেষে যে ১ দেওয়া আছে সেটি হচ্ছে ইনক্রিমেন্ট ভ্যালু (increment valuc)। লুপের ভিতরের স্টেটমেন্টগুলো (লাইন ৩০ থেকে লাইন ৭০) যতবার লুপ চলবে ততবার execute হবে।

nested লুপের উদাহরণ :

```
100 FOR A = 1 TO 10
::
150 FOR B = 1 TO 5
:
200 NEXT B
:
250 NEXT A
```

FOR স্টেটমেন্টের ফরম্যাট (format) :

FOR loop variable = starting value / variable / expression TO end valuc / variable / expression

[STEP increment value / variable / expression]

FOR স্টেটমেন্টের কিছু উদাহরণ :

FOR X=1 to 10	X এর মান 1 দিয়ে লুপ শুরু হয় এবং শেষ হয় যখন X এর মান 10—লুপটি চলে ১০ বার। increment value এক্ষেত্রে ১ (যেহেতু কোন নির্দিষ্ট মান বলে দেওয়া নেই)।
FOR N = 5 TO 100 STEP 5	5 দিয়ে লুপ শুরু হয় শেষ হয় ১০০তে এসে, প্রতিবারে লুপ ভেরিয়েবল X এর মান বাড়ে 5 করে, লুপ চলে 20 বার।
FOR A = B TO C STEP D	লুপ শুরু হয় ভেরিয়েবল B এর মান দিয়ে, শেষ হয় ভেরিয়েবল C এর মান দিয়ে, প্রতিবার লুপ ভেরিয়েবলের মানের সাথে D এর মান যোগ হয়।
FOR X = A TO B + 10	লুপ শুরু হয় ভেরিয়েবল A এর মান দিয়ে আর শেষ হয় যখন লুপ ভেরিয়েবলের মান হয় B + 10 এর সমান।

NEXT স্টেটমেন্টের কিছু উদাহরণ :

NEXT X	X এর লুপ শেষ করে (X লুপ ভেরিয়েবেল)
NEXT A, B	প্রথমে A এর লুপ এবং তারপর B এর লুপ শেষ করে।

FOR ও NEXT ব্যবহার করে এখানে একটি প্রোগ্রাম দেওয়া হল যা 1 থেকে 19 পর্যন্ত সব বেজোড় সংখ্যা প্রিন্ট করে।

Prog. 5A.

```
10 CLS
20 PRINT "ODD NUMBERS FROM 1 TO 19"
30 FOR N = 1 TO 19 STEP 2
40 PRINT N
50 NEXT N
60 END
```

আর একটি প্রোগ্রাম দেওয়া হল যা কোন একটি সংখ্যার multiplication table (নামতা) প্রিন্ট করবে। সংখ্যাটি প্রোগ্রামের ইনপুট হিসেবে নেওয়া হবে।

Prog. 5B

```
10 CLS
20 PRINT "MULTIPLICATION TABLE"
30 INPUT "Multiplication table cn what number : ";N
40 FOR A = 1 TO 5
50 PRINT N * A
60 NEXT A
70 END
```

উপরের প্রোগ্রামের আউটপুট আরও সুন্দর করার জন্য কিছু পরিবর্তন সাধন করা যেতে পারে।

যেমন, স্ক্রীনে প্রম্পট (prompt) এবং "MULTIPLICATION TABLE" এই হেডিং এর মাঝে একটি ফাঁকা লাইন ঢোকাতে চাইলে লিখুন।

```
15 PRINT
```

আবার টেবিলের স্টাইল যদি খানিকটা বদলাতে চান, তাহলে লাইন 50 এর বদলে লিখুন, `50 PRINT N ; "X"; A; "=" ; N*A`

এই পরিবর্তনগুলো করার পর যদি আপনি prog. 5B. RUN করান তাহলে আউটপুট হবে এরকম—

```
RUN
MULTIPLICATION TABLE
Multiplication table on what numbers : 3
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
OK
```

প্রোগ্রামের লাইনগুলোকে সমব্যবধানে পুনর্বিন্যাস্ত করতে চাইলে আপনি RENUM কম্যাণ্ডটি ব্যবহার করতে পারেন। স্ক্রীনে একটি ফাঁকা লাইনে RENUM কথাটি টাইপ করে এন্টার চাবিটি চাপুন। এখন প্রোগ্রাম লাইনগুলো 10 ব্যবধানে পুনর্বিন্যাস্ত হবে।

RENUM কমাণ্ডের ফরম্যাট :

RENUM [start, interval]

Start value হচ্ছে সেই সংখ্যাটি যা হবে প্রোগ্রামের প্রথম লাইন নম্বর এবং interval value হচ্ছে সেই সংখ্যা মান, যার ব্যবধানে আপনি লাইন নম্বরগুলোকে পরিবর্তন করতে চান।

READ/DATA স্টেটমেন্ট

প্রোগ্রামে ব্যবহৃত ডাটা প্রোগ্রামে দেবার জন্য DATA স্টেটমেন্ট ব্যবহার করা হয়। DATA স্টেটমেন্টের মাধ্যমে যে ডাটাগুলো প্রোগ্রামে দেওয়া হয়, সেগুলো পড়ে ভেরিয়েবলে রাখার জন্য READ স্টেটমেন্ট ব্যবহার করা হয়।

READ স্টেটমেন্টের ফরম্যাট :

READ variable [variable, varibale, ...]

উদাহরন :

```
READ N
READ A, B
READ NAMES, ADDRS, AGE, TEL
```

DATA স্টেটমেন্টের ফরম্যাট :

DATA data-item [, data-item, data-item ...]

উদাহরণ :

```
DATA 6
DATA 14, 18
DATA Jolly Nasreen, "144, 145 ABC Road", 29
```

প্রোগ্রামের যে কোন জায়গাতেই READ/DATA স্টেটমেন্ট ব্যবহার করা যায়। READ স্টেটমেন্ট DATA স্টেটমেন্ট থেকে ক্রমানুসারে ডাটা পড়ে নিয়ে তা উল্লিখিত ভেরিয়েবলে রাখে। DATA স্টেটমেন্টের প্রতিটি ডাটা কমা (,) চিহ্ন দিয়ে আলাদা করা হয়। আপনার প্রোগ্রামে একাধিক DATA স্টেটমেন্ট থাকতে পারে। সবচেয়ে কম নম্বরযুক্ত লাইনের DATA স্টেটমেন্ট থেকে READ স্টেটমেন্ট ডাটা পড়া আরম্ভ করে (একের পর এক, DATA স্টেটমেন্ট এর লাইন নম্বরের উচ্চ ক্রমানুসারে)। এক্ষেত্রে মনে রাখবেন READ এ উল্লিখিত ভেরিয়েবল এর ধরন (type) ও DATA স্টেটমেন্টের ডাটার ধরণ একই হতে হবে। READ স্টেটমেন্টে যে ধরণের ভেরিয়েবল উল্লেখ করা হয়েছে (READ AGE, NAMES) তা যদি ডাটা স্টেটমেন্টের ডাটার ধরণের (DATA ALI, 23) সাথে না মিলে তাহলে 'SYNTAX ERROR' দেখা দেবে।

প্রোগ্রামে READ/DATA স্টেটমেন্টের ব্যবহার কিছু উদাহরণের সাহায্যে এখানে দেখানো হল।

উদাহরণ ১
```
50 READ A, B, C
60 PRINT A + B+ C
70 END
80 DATA 53, 27, 20
```

উদাহরন ২
```
50 DATA 1, 5
60 READ S, E
70 FOR X = S TO E
80 READ AGE : PRINT X; AGE
90 NEXT X
100 END

200 DATA 23, 31, 29, 34, 26
```

প্রথম উদাহরণে লাইন 50 ৩টি ডাটা পড়ে এবং লাইন 60 এই ৩টি ডাটার সমষ্টি প্রিন্ট করে। ডাটা 53, 27 ও 20 যথাক্রমে ভেরিয়েবল A, B ও C তে রাখা হয়। দ্বিতীয় উদাহরণে লাইন 60 লাইন 50 এর DATA স্টেটমেন্ট থেকে ডাটা 1 ও 5 পড়ে ভেরিয়েবল S ও E তে রাখে। লাইন 80 এর READ স্টেটমেন্ট লাইন 200 এর DATA স্টেটমেন্ট থেকে ডাটা পড়ে নেয়। লাইন 60 এর READ স্টেটমেন্ট যে ডাটাগুলো পড়ে তা পরবর্তী লাইনের FOR স্টেটমেন্টের starting ও end value হিসেবে ব্যবহার করা হচ্ছে। FOR লুপটি ৫ বার চলবে। যেহেতু লাইন ৮০ এর READ স্টেটমেন্টটি লুপের ভেতর অবস্থিত। তাই এটি ৫ বার ডাটা (একের পর এক) পড়বে। একই লাইনের প্রিন্ট স্টেটমেন্টটি X ও AGE এর মান প্রিন্ট করবে। নীচের প্রোগ্রামটি ১০টি ধনাত্মক সংখ্যার মধ্য থেকে বৃহত্তম সংখ্যাটি প্রিন্ট করে।

Prog. 6A
```
10 H = 0
20 FOR X = 1 TO 10
30 READ N
40 IF N > H THEN H = N
50 NEXT X
60 PRINT "The highest number is: ";H
70 END

80 DATA 4, 2, 7, 12, 1, 16, 23, 24, 22, 9
```

আপনি হয়তো কোন ব্যক্তির নাম জেনে তার সম্পর্কিত অন্যান্য তথ্য বের করতে চান এবং এই তথ্য খোঁজবার দায়িত্বটা কমপিউটারকে দিতে চান, এই ধরণের কাজ করে এমন একটি বেসিক প্রোগ্রাম এখানে দেয়া হলো। প্রোগ্রামটি আপনাকে READ/DATA স্টেটমেন্টের ব্যবহার সম্পর্কেও কিছু ধারণা দেবে।

Prog. 6B
```
10 CLS
20 "Enter the Name to search : "; NS
40 F=O
50 READ NAMES,ADDRS,AGE,TEL
60 IF NS<>NAMES THEN GOTO 130
70 F=1
80 PRINT "NAME      : ";NAMES
90 PRINT "ADDRESS : " ; ADDRS
100 PRINT "AGE       : " ; AGE
110 PRINT "PHONE    : " ;TEL
120 PRINT
130 NEXT R
140 IF F=0 THEN PRINT "NOT FOUND"
150 DATA HASSAN,12/B RIVER SIDE,23,327678

160 DATA JOLLY,154 KINGS SHOP,30,324887
```

উপরের প্রোগ্রামটিতে লাইন 20 user এর কাছ থেকে একটি নাম ইনপুট হিসেবে গ্রহণ করে। লাইন 40 ও 130 এর FOR ও NEXT স্টেটমেন্ট দুটি দিয়ে একটি লুপ তৈরি করা হয়েছে যা পাঁচ বার চলবে (৫টি নামের জন্য)। লুপের ভেতরে লাইন 50 এর READ স্টেটমেন্ট DATA স্টেটমেন্ট থেকে ডাটা পড়ে নেয়। প্রতি execution এ READ স্টেটমেন্টটি চারটি ডাটা পড়ে। NAMES, ADDRS, AGE ও TEL এই চারটি ভেরিয়েবলে রাখে। লাইন 160 এর শর্তযুক্ত স্টেটমেন্ট প্রোগ্রামটিকে লাইন ১৩০ এ পাঠিয়ে দেবে যদি NAMES ও N$ এর মান সমান না হয়। যখন NAMES এর মান (যা DATA স্টেটমেন্ট থেকে পড়া হয়েছে) ও N$ এর মান (যা user-এর কাছ থেকে নেয়া হয়েছে) সমান হবে তখন শর্তটি মিথ্যা হয়ে যাবে এবং প্রোগ্রাম লাইন ৭০ এ গিয়ে F ভেরিয়েবলের মান ১ করে দেবে। F এর মান 1 হলে বুঝতে হবে যে user এর দেয়া নামটি ডাটা লিস্টে পাওয়া গেছে। লাইন 80 থেকে 110 name, address ইত্যাদি প্রিন্ট করে। লাইন 120 একটি ফাঁকা লাইন প্রিন্ট করে। লাইন 140 "NOT FOUND" এই messageটি প্রিন্ট করবে যদি সবগুলো (এক্ষেত্রে ৫টি) নাম ডাটা লিস্টের নামগুলোর সাথে না মিলে। এই প্রোগ্রামে শুধু দুই ব্যক্তির তথ্য দেয়া হয়েছে (লাইন ১৫০ ও ১৬০)। আরও তিন ব্যক্তির তথ্য প্রোগ্রামে সংযোগ না করলে আপনি 'OUT OF DATA' এই ত্রুটির সম্মুখীন হবেন। এখন আপনি প্রোগ্রামটি ভাল করে পড়ে নিজের প্রয়োজন অনুযায়ী পরিবর্তন করে নিন।

প্রিন্ট করবে, অন্যথায় 'Try again' কথাটি প্রিন্ট করবে।

Prog. 6C
```
10 CLS
20 FOR  T=1 TO 5
30 READ A,B
40 PRINT "What is the sum of ";A;"and";B
50 INPUT "Enter the correct sum " ; S
60 IF A+B <> S THEN PRINT "Try again...":GOTO 40
70 PRINT "Good"
80 NEXT T
90 END

100 DATA 45,47,69,93,103,234,859,547
```

প্রোগ্রামটি RUN করালে আপনি দেখবেন:

```
RUN
What is the sum of 4 and 5
Enter the correct sum ? 7
Try again ...
What is the sum of 4 and 3
Enter the correct sum ? 9

Good
```

(এভাবে মোট ৫ বার প্রিন্ট করবে ৫ জোড়া সংখ্যার জন্য)

বেসিক প্রোগ্রাম সেভ (save) এবং লোড (load) করার উপায় :

আপনি আপনার বেসিক প্রোগ্রাম ডিস্কে সেভ (সংরক্ষণ) করতে পারেন বেসিকের SAVE কমাণ্ড ব্যবহার করে। আবার সেই প্রোগ্রামটি পরবর্তীতে ডিস্ক থেকে পড়ে নিয়ে মেমোরিতে লোড করতে পারেন LOAD কমাণ্ড ব্যবহার করে। আপনি যখন কোন প্রোগ্রাম সেভ করবেন, তখন বেসিক প্রোগ্রামের নামের শেষে 'BAS' ফাইল এক্সটেনশনটি যোগ করে দেবে। SAVE এবং LOAD টাইপ না করে যথাক্রমে F3 ও F4 ফাংশন কী গুলো চেপেও আপনি একই ফল পাবেন।

যে প্রোগ্রাম নিয়ে আপনি কাজ করছেন, সেটি যদি PROG3B নামে B ড্রাইভে সেভ করেত চান, তাহলে একটি খালি লাইনে SAVE "PROG4B" টাইপ করে এন্টার চাবিটি চাপুন। ড্রাইভ B থেকে PROG3B প্রোগ্রামটি মেমোরিতে নিয়ে আসতে চাইলে LOAD "PROG3B" টাইপ করে এন্টার চাবিটি চাপুন। একবার কোন প্রোগ্রাম মেমোরিতে লোড করে নেওয়ার পর আপনি LIST, RUN এই কমাণ্ডগুলো চালাতে পারেন অথবা প্রোগ্রামে যে কোন পরিবর্তন সাধন করতে পারেন।

SAVE এর ফরম্যাট :

```
SAVE "[drive-name [path]] filename"
```

উদাহরণ :

SAVE "b:test"	কারেন্ট (বর্তমানে যেটি মেমোরিতে আছে) প্রোগ্রামটি B ড্রাইভে 'TEST' নামে সেভ করা হবে।
SAVE "prog1"	কারেন্ট প্রোগ্রামটি 'PROG' নামে যে ড্রাইভ/পাথ (path) থেকে বেসিক লোড করা হয়েছে সেখানে সেভ করা হবে।

LOAD এর ফরম্যাট:

```
LOAD "[drive-name[path]] filename"
```

উদাহরণ :

LOAD "b:test"	TEST. BAS প্রোগ্রামটি ড্রাইভ B থেকে লোড করা হবে।
LOAD "Prog1"	যে ড্রাইভ/পাথ থেকে বেসিক লোড করা হয়েছে সেখান থেকে PROG1. BAS প্রোগ্রামটি লোড করা হবে।

সত্যি বলতে কি যারা সবেমাত্র বেসিক শেখা শুরু করেছেন তাদের জন্য শুধু পড়ার মাধ্যমে প্রোগ্রামিং সম্পর্কে ধারণা গড়ে ওঠা খুব সোজা নয়। তবুও ভালভাবে চেষ্টা ও চর্চা করলে আপনি হয়তো বিভিন্ন সমস্যা সমাধানের জন্য ছোট ছোট প্রোগ্রাম লিখতে পারবেন। কোন সমস্যা সমাধানের জন্য পাঠকদের প্রতি আমার পরামর্শ হল প্রথমে সোজা ভাষায় সমস্যাটি পরিষ্কারভাবে চিহ্নিত করুন, তারপর প্রোগ্রাম ফ্লো (flow) এঁকে প্রোগ্রামের প্রতিটি ধাপ নির্দেশ করুন। সবশেষে সঠিক কীওয়ার্ড ব্যবহার করে প্রোগ্রামটি লিখুন।

বেসিক শেখার এই পর্বে যেসব কীওয়ার্ড ও বিষয় আলোচিত হয়েছে তা বেসিক প্রোগ্রামিং এর অতি ক্ষুদ্র একটি অংশমাত্র। বেসিক প্রোগ্রামিং এর বিশালত্ব সম্পর্কে এই সংক্ষিপ্ত আলোচনা থেকে কোন ধারণাই করা যাবে না। এই আলোচনার উদ্দেশ্য হচ্ছে পাঠকদের বেসিক এর সাথে পরিচয় করিয়ে দেয়া।

এই সংখ্যায় আলোচিত প্রধান বিষয়গুলো হচ্ছে:

user এর কাছ থেকে ডাটা ইনপুট নেয়ার জন্য INPUT স্টেটমেন্টের ব্যবহার; IF স্টেটমেন্ট ব্যবহারের মাধ্যমে প্রোগ্রামে শর্ত আরোপ করা ও শর্তানুযায়ী কার্য সম্পাদন করা। কতগুলো বেসিক স্টেটমেন্ট পুনঃপুনঃ execute করার জন্য FOR ... NEXT স্টেটমেন্ট ব্যবহার করে লুপ তৈরি করা; RENUM কমাণ্ড ব্যবহার করে প্রোগ্রামের লাইন নম্বর একটি ক্রমানুযায়ী পুনর্বিন্যাস্ত করা; READ ... DATA স্টেটমেন্ট ব্যবহার করে প্রোগ্রামে ডাটা পড়া; SAVE ও LOAD কমাণ্ড দিয়ে প্রোগ্রাম সেভ ও লোড করা। এসব কীওয়ার্ডগুলোর ব্যবহার পাঠকদের দেখানোর জন্য উদাহরণ স্বরূপ কিছু প্রোগ্রামও এখানে সংযোজিত হয়েছে।

অনুলিখনঃ কাজী ইফ্ফাত হক