Creating a procedural computer program using COBOL

Level 2

Notes

for

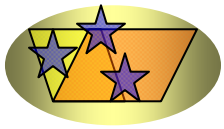
City & Guilds

7540 Unit 005

Compatible with Micro Focus® Net Express 5.0 COBOL compiler

Tench Computing Ltd
Pines
Glendale Road
Burgess Hill
West Sussex
RH15 0EJ

Web address: www.tenchcomputing.co.uk
Email address: jtench@globalnet.co.uk



About the author: *Jackie Tench MSc, ACIB, Cert Ed*

Jackie started her working career in branch banking with the Midland Bank (now HSBC) and was transferred to their Computing Department after achieving 100% in their ability test for programmers. She then worked for more than a decade in this department and was one of the first women to achieve a junior management grade at the age of 21. She attended a significant number of IBM programming training courses during her time there.

Jackie was the first woman to pass the ACIB (Associate Chartered Institute of Bankers) examinations in the Midland Bank (HSBC) and the youngest person at 21 years of age.

Jackie then left to raise a family but still found time to teach part-time at a college in Sheffield and to obtain a MSc in Computing and a Cert Ed in teaching.

When her children were old enough Jackie returned to work full-time and was a Senior Lecturer in Software Engineering and Computer Studies at a college in Brighton for nearly 10 years teaching all levels up to and including HND.

Therefore, Jackie has considerable business knowledge and qualifications plus wide experience in practical computing and training – covering areas such as structured design, analysis, coding, testing and implementing software applications plus training students to fulfil an important role in the computer industry.

Jackie has worked as a consultant for several blue chip companies and examination boards using her software engineering and educational training skills and is now one of the foremost experts in computing with an extensive knowledge of programming languages and applications.

Contents Page

[This page is intentionally blank]

Structure of a COBOL program

A COBOL program consists of four **divisions**.

Identification division

This is used for documentation purposes on microcomputers and the entries are optional. They are included to provide compatibility with other more powerful versions of COBOL. The PROGRAM-ID entry is usually included in order to identify a program.

Format

Square brackets indicate the entry is optional.

Lower case items must be supplied by the programmer.

```
IDENTIFICATION DIVISION.
[PROGRAM-ID. program-name.]
[AUTHOR. author name.]
[INSTALLATION. installation-name.]
[DATE-WRITTEN. date.]
[DATE-COMPILED. date.]
[SECURITY. comment.]
```

Environment division

This specifies the environment in which the program is to be run.

It consists of 2 sections:-

CONFIGURATION SECTION.

The entries in this section, with one exception, are for documentation purposes only and are optional. They are included to provide compatibility with other more powerful versions of COBOL.

The exception is the SPECIAL-NAMES paragraph; this is used with the clause CONSOLE IS CRT. This causes the default device CONSOLE, used in the ACCEPT and DISPLAY statements, to be changed to enable data to be accepted from and displayed on the **screen** for microcomputers.

Another clause which may be required in the SPECIAL-NAMES paragraph is the CURRENCY SIGN IS "£" this allows the £ sign to be used in PICTURE clauses in the data division to represent the currency symbol.

INPUT-OUTPUT SECTION.

This section is required if any files are being used by the program. There are two paragraphs in this section but only the FILE-CONTROL paragraph is normally required. Each file used by the program, including printer files, will need a SELECT statement which associates the internal filename with the required external medium and external filename.

Format

Square brackets indicate the entry is optional.

Lower case items must be supplied by the programmer.

```
        ENVIRONMENT DIVISION.
        CONFIGURATION SECTION.
        [SOURCE-COMPUTER. computer-name.]
        [OBJECT-COMPUTER. computer-name.]
        SPECIAL-NAMES. CONSOLE IS CRT.
[       INPUT-OUTPUT SECTION.
 [      FILE-CONTROL.                    ]
 [          file control entries.        ]
 [      I-O-CONTROL.                     ]
 [          input output control entries.]
]
```

Data division

There are 3 sections in the data division. If any section is not required it can be omitted.

FILE SECTION.

This defines the structure of all the input and output files used by the program. Each file is defined by a file description (FD) entry and one or more record descriptions.

WORKING-STORAGE SECTION.

This contains all the temporary data items used by the program while it is running but which are not part of external files.

LINKAGE SECTION.

This section is used to define data items which are to be passed between programs.

Format

Square brackets indicate the entry is optional.

Lower case items must be supplied by the programmer.

```
        DATA DIVISION.
[       FILE SECTION.
        file description
        record description(s)
        file description
        record description(s)
]
[       WORKING-STORAGE SECTION.
        single item descriptions
        record descriptions
]
[       LINKAGE SECTION.
        single item descriptions
        record descriptions
]
```

Procedure division

This contains the program instructions that will be executed and will operate on the data described in the data division. The physical end of a program is the last program instruction in the source code. The logical end of a program is the last program instruction to be executed e.g. STOP RUN which is not necessarily the last program instruction in the source code.

Format of a COBOL program

Lower case items must be supplied by the programmer.

Since sequence numbers are not used the source records start in column 8.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. program-name.
AUTHOR. author name.
DATE-WRITTEN. date.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. computer-name.
OBJECT-COMPUTER. computer-name.
SPECIAL-NAMES.
        CONSOLE IS CRT.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT statements with details of files used.
DATA DIVISION.
FILE SECTION.
FD file-name.
01 record-name.
        02 data-name1          PIC format.
        02 data-name2          PIC format.
                (etc.)
WORKING-STORAGE SECTION.
77 data-name3                  PIC format.
77      .
77      .
PROCEDURE DIVISION.
PARA-1.
        .


PARA-2.


        STOP RUN.
PARA-3.
```

Questions 2

1. Name the 4 DIVISIONS used in a COBOL program?

2. What data items are declared in the WORKING-STORAGE SECTION?

3. What does the PROCEDURE DIVISION contain?

DATA DIVISION

Data description entries

Data description entries are used to describe the data that is to be stored and used by the program.

General Format

```
level-number   { data-name-1 }
               { FILLER      }
      [ REDEFINES data-name-2 ]

[ { PICTURE } IS character-string ]
  { PIC     }

[      [ USAGE IS ] { COMPUTATIONAL   } ]
                    { COMP            }
                    { COMPUTATIONAL-3 }
                    { COMP-3          }
                    { DISPLAY         }

[      [ SIGN IS ] { LEADING  } [SEPARATE CHARACTER] ]
                   { TRAILING }

[ { JUSTIFIED } RIGHT ]
  { JUST      }

      [ BLANK WHEN ZERO ]
      [ VALUE  IS literal ]  .
```

Square brackets indicate that the clause is optional.

Curly brackets indicate that a choice must be made from the items included within the brackets.

Lower case items must be supplied by the programmer.

Upper case items are COBOL reserved words. The underlined portion is mandatory if the item is selected from an optional list.

User defined names

Data names are supplied by the programmer and are user defined names.

A user defined name is one that the programmer supplies to name data items, files or paragraphs. It must not be more than 30 characters in length and may consist of alphabetic and numeric characters plus a hyphen (i.e. no special characters). The first character must be alphabetic and the hyphen, if used must not be the first or last character.

A user defined name may not be used for more than one data item, file-name or paragraph i.e. it must be unique.

COBOL reserved words cannot be used as user defined words. The majority of reserved words used in COBOL do not have hyphens, therefore, if user defined names are used which have hyphens there is less chance of using a reserved word by mistake.

Valid names	**Invalid names**
M-NAME	-NAME *Hyphen at beginning*
MAST-FL	MASTFL- *Hyphen at end*
AREA-2	AREA *Reserved word*
M-ADDR1	&M-ADDR1 *Invalid character*

Since COBOL is meant to be easily read and understood user defined words should be meaningful, but not too long.

Level numbers

The level number may be any number from 01 to 49 or 77 or 88.

If a data item is to be used on its own and is not part of a group data item the level number used is 77.

```
77   COUNT-1                PIC 99.
```

We have defined this data item with the data name COUNT-1 and specified with the PICTURE clause (PIC is an abbreviation for PICTURE) that it is numeric (9) and that it is 2 digits in length (by using 99).

All 77 data items are elementary items and therefore must have a PICTURE clause.

For data items that are grouped together the level numbers must be in the range 01 - 49.

If a date was to be defined the level number 77 could be used.

```
77   DATE-IN                PIC 9(8).
```

DATE-IN has been defined as 8 numeric digits and the date in this field could be stored as DDMMYYYY where DD is the day, MM is the month and YYYY is the year. Note the use of 9(8) rather than PIC 99999999, the number in brackets indicates the number of times the character occurs, this saves writing out each character in the character-string separately.

To access the DD, MM and YYYY parts of the date in a program it would have to be defined as follows:-

```
01   DATE-IN.
     02 DD                  PIC 99.
     02 MM                  PIC 99.
     02 YYYY                PIC 9999.
```

By defining the date as above; DATE-IN is a group item and DD, MM, YYYY are elementary items (i.e. they have no data items subordinate to them) which make up the group item. The group item does not have a PICTURE clause only elementary items have a PICTURE clause.

DATE-IN		
DD	MM	YYYY

In the program DATE-IN will reference the whole of the date (8 digits). Alternatively, DD will reference just the days in the date (2 digits) or MM will reference the month in the date (2 digits) or YYYY will reference the year in the date (4 digits).

If the date was to be stored as DD/MM/YYYY then it could be defined as:-

```
01 DATE-IN.
      02 DD          PIC 99.
      02 FILLER      PIC X.
      02 MM          PIC 99.
      02 FILLER      PIC X.
      02 YY          PIC 9999.
```

In the program DATE-IN will reference the whole of the date (10 characters). DD, MM or YYYY can be used to reference individual items which make up the date.

FILLER is used to name a field that will not be referenced by the program. If the / character in the FILLER field needs to be referenced then it must be given a data name as FILLER cannot be referenced by the program instructions.

Level number 88 is used to define condition-names for a variable i.e. values that it can take.

```
01 PAGE-SWITCH PIC X.
      88 SWITCH-ON       VALUE 'Y'.
      88 SWITCH-OFF      VALUE 'N'.
```

The variable PAGE-SWITCH can take the values Y or N. The value can be tested using an IF statement. (IF SWITCH-ON THEN …).

PICTURE clause

{ PICTURE / PIC } IS character-string

The IS is normally omitted.

PIC is an abbreviation for PICTURE.

The PICTURE clause can only be used for elementary data items.

There are **four** main categories of data that can be described with a PICTURE clause: **alphabetic**; **numeric**; **alphanumeric**; **numeric edited**.

Alphabetic data

The PICTURE character-string for alphabetic data can contain the symbol A.

Each A represents a character position which can contain only a letter of the alphabet or a space.

```
77   NAME-1                PIC A(15).
```

The length of the field NAME-1 is 15 characters and each character can be alphabetic or a space.

Alphanumeric data

The PICTURE character-string for alphanumeric data consists of X's.

```
77   NAME-IN               PIC X(13).
```

The length of the field NAME-IN is 13 characters. The contents of NAME-IN can consist of any characters in the computers character set.

Numeric data

The PICTURE character-string can contain 9, S or V. The maximum number of digits is 18.

9 Each 9 represents a character position which holds a numeric digit.

```
77   COUNT-1               PIC 999.
```

COUNT-1 holds a data item which has 3 numeric digits.

S The letter S is used to indicate the presence of a sign. It must be the first character in the character-string.

It is not counted in the length of the data item unless a SIGN clause is used which specifies SEPARATE character.

```
77   AMOUNT-1              PIC S9(5).
```

AMOUNT-1 holds a data item which has 5 numeric digits and can be positive or negative. The size of the field is 5 bytes.

V The letter V is used to indicate the location of the assumed decimal point. It can appear only once in a character-string and is not counted in the size of the data item.

```
77   BALANCE-1             PIC 9(4)V99.
```

BALANCE-1 can hold a number of up to four numeric digits before the decimal point and two after. The size of the field is 6 bytes.

```
77   BALANCE-2             PIC S9(5)V9(4).
```

BALANCE-2 can hold a number which is positive or negative and has up to 5 numeric digits before the decimal point and 4 after. The size of the field is 9 bytes.

```
77  BALANCE-3          PIC 9(5).
```

BALANCE-3 can hold an integer number of up to 5 numeric digits. The V is not required because when it is omitted the number is assumed to be an integer.

The assumed decimal point is used to align numeric data items for numeric calculations.

If BALANCE-1 contains 0015V90 and BALANCE-2 contains 02346V4320 and both numbers are positive. If BALANCE-1 is added to BALANCE-2.

```
 0015V90          BALANCE-1
02346V4320        BALANCE-2
------------------
02362V3320        Result in BALANCE-2
```

The numeric data items would be aligned, as for normal additions, using the assumed decimal point V.

> Care must be taken when specifying the length of numeric data items. If the field is not large enough digits will be lost.

If BALANCE-1 contains 4315V50 and BALANCE-3 contains 06432 and BALANCE-3 is added to BALANCE-1 with the result in BALANCE-1.

```
 4315V50          BALANCE-1
 06432            BALANCE-3
----------------
(1)0747V50        Result in BALANCE-1
```

The answer is 10747V50, but BALANCE-1 has only 4 digits before the assumed decimal point, therefore the most significant digit will be lost and BALANCE-1 will contain 0747V50. This means that BALANCE-1 contains incorrect data.

Numeric edited data

The maximum number of digits that can be represented is 18.

Numeric edited data is used for describing numeric fields which are to be printed or displayed on the screen. Numeric edited data **cannot** be used in numeric calculations but it can be used after the GIVING option in arithmetic statements.

Fixed insertion editing

This allows characters to be inserted into the numeric field. The length of the data item will include the inserted characters. The characters to be inserted will appear in the same positions in the field as they appear in the character-string. Only one currency symbol and only one of the editing sign control symbols can be used in a given PICTURE character-string. A full stop is used to represent the decimal point and may occur only once in the character-string. If present it will be used to align the assumed decimal point when a numeric data item is moved to the numeric edited data item.

Fixed insertion of the £ currency symbol. If the £ sign is used as the currency symbol then the CURRENCY SIGN IS "£" clause must be included in the SPECIAL-NAMES paragraph. The currency symbol must be the leftmost character in the PICTURE character-string. The value moved to the field is aligned and if the value does not fill the field then leading or trailing zeros are inserted.

```
77  TOTAL-1        PIC £999.
```

If the value 55 is moved to TOTAL-1 then TOTAL-1 will contain

£055

If the value 133 is moved to TOTAL-1 then TOTAL-1 will contain

£133

```
77  NUM1            PIC £9,999.99.
```

If the value 3463V45 is moved to NUM1 then NUM1 will contain

£3,463.45

If the value 365V2 is moved to NUM1 then NUM1 will contain

£0,365.20

If the value 54321V54 is moved to NUM1 then NUM1 will contain £4,321.54, the most significant digit having been lost since the field was not large enough.

```
77  NUM2            PIC £99,999.
```

If the value 4563 is moved to NUM2 then NUM2 will contain

£04,563

If the value 50497 is moved to NUM2 then NUM2 will contain

£50,497

If the value 20V2 is moved to NUM2 then NUM2 will contain

£00,020

The least significant digit has been lost since the PICTURE character-string does not specify any digits after the decimal point.

Fixed insertion of the sign character. The sign character can be at the beginning or end of the character-string.

If the + character is used then if the value moved to the field is positive a + will be inserted, if the value is negative a - will be inserted.

```
77   TOTAL-2          PIC £999.99+.
```

If the value 65 is moved to TOTAL-2 then TOTAL-2 will contain

£065.00+

If the value -43V35 is moved to TOTAL-2 then TOTAL-2 will contain

£043.35-

```
77   NUM3             PIC +999.
```

If the value 555 is moved to NUM3 then NUM3 will contain

+555

If the value -45 is moved to NUM3 then NUM3 will contain

-045

If the - character is used to indicate the position of the sign, then if the value in the field is positive a space will be inserted, if the value in the field is negative a - will be inserted.

```
77   NUM5             PIC 999.99-.
```

If the value 85V21 is moved to NUM5 then NUM5 will contain

085.21b

The character b represents a space character.

If the value -750V5 is moved to NUM5 then NUM5 will contain

750.50-

CR and **DB** can also be used to indicate whether a numeric data item is positive or negative. If CR is used and the data item is positive then 2 spaces will be inserted in place of the CR symbol; if the data item is negative then the CR will be inserted. If DB is used and the data item is positive then 2 spaces will be inserted in place of the DB symbol; if the data item is negative then the DB will be inserted.

```
77   NUM5             PIC £999.99CR.
```

If the value 23V5 is moved to NUM5 then NUM5 will contain

£023.50bb

If the value -125V12 is moved to NUM5 then NUM5 will contain

£125.12CR

```
77   NUM6             PIC 9,999.99DB.
```

If the value 4234V45 is moved to NUM6 then NUM6 will contain

4,234.45bb

If the value -253V5 is moved to NUM6 then NUM6 will contain

0,253.50DB

Floating insertion editing

The currency symbol and the symbols + or - can be used as floating insertion characters. Only one can be used in a given PICTURE character-string.

```
77   NUM4                PIC £££,££9.99.
```

The maximum number which can be stored in NUM4 is 99,999.99 since the £ sign occupies the leftmost position when the other positions contain non-zero digits.

If the value 33V3 is moved to NUM4 then NUM4 will contain

bbbb£33.30

The £ sign is floated up to the first non-zero digit.

```
77  NUM5                PIC ££,£££.99.
```

If the value 2345V25 is moved to NUM5 then NUM5 will contain

£2,345.25

If the value 56443V25 is moved to NUM5 then NUM5 will contain

£6,443.25

The most significant digit is lost since the character-string is not large enough.

If the value V2 is moved to NUM5 then NUM5 will contain

bbbbb£.20

```
77   NUM6                PIC ££,£££,£££.
```

If the value zero is moved to NUM6 then NUM6 will contain

bbbbbbbbbb

```
77   NUM4                PIC +,++9.99.
```

If the value 21V56 is moved to NUM4 then NUM4 will contain

bb+21.56

```
77   NUM5                PIC --9.
```

If the value 8 is moved to NUM5 then NUM5 will contain

bb8

If the value -4 is moved to NUM5 then NUM5 will contain

b-4

Suppression and replacement editing

Leading zeros can be suppressed by using Z or * as suppression characters in the PICTURE character-string. If Z is used, the replacement character is a space and if the asterisk is used, the replacement character is a *. The Z and the * are mutually exclusive i.e. cannot be used in the same character-string.

```
77   NUM2                PIC ZZ,ZZ9.99.
```

If the value 6712V45 is moved to NUM2 then NUM2 will contain

b6,712.45

If the value 55645V3 is moved to NUM2 then NUM2 will contain
    55,645.30
If the value V5 is moved to NUM2 then NUM2 will contain
    bbbbb0.50

    77 NUM3   PIC ZZZ.ZZ.
If the value zero is moved to NUM3 then NUM3 will contain
    bbbbbb
If the value 40 is moved to NUM3 then NUM3 will contain
    b40.00

    77 NUM4   PIC £*****.99.
If the value 4235V23 is moved to NUM4 then NUM4 will contain
    £*4235.23

    77 NUM5   PIC ***.**.
If the value 30 is moved to NUM5 then NUM5 will contain
    *30.00
If the value zero is moved to NUM5 then NUM5 will contain
    ***.**

VALUE clause

This clause is used to set an initial value for a data item in the WORKING-STORAGE SECTION. The VALUE clause can only be used for an elementary data item.

    77 PI   PIC 9V999 VALUE 3.142.
PI would be set to 3.142 at the start of the program.

    77 TOTAL-1   PIC 999 VALUE 10.
TOTAL-1 would be set to 10 at the start of the program.

    77 HEAD1   PIC X(11) VALUE "REPORT LIST".
HEAD1 would contain REPORT LIST at the start of the program.

    77 EQUALS-LINE   PIC X(47) VALUE ALL "=".
EQUALS-LINE would contain 47 ='s. Using ALL saves putting in 47 = symbols separately.

Questions 3

1. What are the four main categories of data that can be described with a PICTURE clause?

2. Can numeric calculations be done using a data item which has a numeric edited PICTURE clause?

3. What would be the result of moving the following values to the receiving fields:-

Value	**Receiving Field**
a) 543	PIC 9(4)
b) -99	PIC S9(3)
c) 425V32	PIC 9(3)V99
d) 562V45	PIC 9(4)V9
e) 43	PIC £999
f) 5115V54	PIC £99,999.99
g) 44V22	PIC £9,999.999
h) +15V23	PIC £999.99+
i) -250V55	PIC £99.99+
j) 10V10	PIC 999.99-
k) -354V	PIC 999-
l) 100V3	PIC £££,££9.99
m) 2504	PIC £££,££9.99
n) 5543V5	PIC ZZ,ZZ9.99
o) 269	PIC Z,ZZ9
p) 326V45	PIC £****.99

REDEFINES clause

This clause allows the same storage area to be described by different data description entries in the WORKING-STORAGE SECTION. Since the same area of storage is being used the record with the REDEFINES clause must not be longer than the original record but may be shorter if the level number is 01. If the level number is not 01 then the group item with the REDEFINES clause must be the same length as the original group item.

It must not be used in level 01 entries in the FILE SECTION as multiple records in a file always occupy the same storage area.

The following program describes the record SCREEN-OUT which occupies 560 bytes of storage (calculated by adding up all the PICTURE characters). In SCREEN-OUT all the prompts which are to appear on the screen are described. (See Report Layout Sheet 1).

The record SCREEN-IN redefines SCREEN-OUT and is used to describe the areas on the screen which can have data entered into them. These are the areas blocked in on Report Layout Sheet 1.

SCREEN-OUT and SCREEN-IN occupy the same 560 bytes of storage but have different descriptions for this area.

Key in the following program as REDEF.CBL, the comment lines (*) can be omitted and save it on disk. Compile the program and if there are any syntax errors make the necessary corrections and recompile the program. Then run the program.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SAMPLE-REDEFINES.
 AUTHOR. J TENCH.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES. CONSOLE IS CRT.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*This record describes the prompts which are to be displayed
*on the screen
 01 SCREEN-OUT.
     02 FILLER                  PIC X(13).
     02 PROMPT-1                PIC X(33)
  VALUE "NAME            <               >".
     02 FILLER                  PIC X(127).
     02 PROMPT-2                PIC X(48)
  VALUE "ADDRESS LINE 1  <                              >".
     02 FILLER                  PIC X(112).
     02 PROMPT-3                PIC X(48)
  VALUE "ADDRESS LINE 2  <                              >".
     02 FILLER                  PIC X(112).
     02 PROMPT-4                PIC X(48)
  VALUE "ADDRESS LINE 3  <                              >".
     02 FILLER                  PIC X(19).
*This record uses the same area of storage as SCREEN-OUT and
```

```
*describes the areas on the screen which are to be used for
*data entry
 01 SCREEN-IN REDEFINES SCREEN-OUT.
      02 FILLER               PIC X(30).
      02 NAME-IN              PIC X(15).
      02 FILLER               PIC X(145).
      02 ADDR1-IN             PIC X(30).
      02 FILLER               PIC X(130).
      02 ADDR2-IN             PIC X(30).
      02 FILLER               PIC X(130).
      02 ADDR3-IN             PIC X(30).
      02 FILLER               PIC X(20).
 PROCEDURE DIVISION.
      DISPLAY SPACES.
*The screen prompts are displayed on the screen at row number
*8 and column number 1
      DISPLAY SCREEN-OUT AT 0801.
*Spaces are moved to SCREEN-IN also moved to SCREEN-OUT since
*SCREEN-IN REDEFINES SCREEN-OUT. This ensures the data entry
*fields are cleared before being displayed on the screen.
      MOVE SPACES TO SCREEN-IN.
      DISPLAY SCREEN-IN AT 0801.
*The record SCREEN-IN is accepted from the screen at row number
*8 column number 1. The only areas accessible to the keyboard
*operator are the data items in SCREEN-IN which have been named.
*FILLER fields are not allowed to be accessed.
      ACCEPT SCREEN-IN AT 0801.
      STOP RUN.
```

The program when run should clear the screen and then display the following prompts, for entering the data, starting at row number 8 and column number 1.

```
NAME            <               >
ADDRESS LINE 1  <                              >
ADDRESS LINE 2  <                              >
ADDRESS LINE 3  <                              >
```

Enter a record into the displayed areas.

Cursor keys can be used to move the cursor within a field. To move to the next field use the Tab key.

To move back a field use Shift and Tab key.

When all the data has been entered press the ENTER or RETURN key. The program should then stop.

```
NAME            <WILLIAMS J     >
ADDRESS LINE 1  <15 SANDYGATE PARK ROAD        >
ADDRESS LINE 2  <BIRMINGHAM                    >
ADDRESS LINE 3  <                              >
```

Using this method a whole record of data can be entered with one ACCEPT statement.

PROGRAMMER: J D TENCH

PROGRAM NAME: REDEF **DATE:** 01/01/99

REPORT LAYOUT SHEET 1

FILENAME:

RECORD NAME: SCREEN

```
         1111111111222222222233333333334444444444555555555566666666667777777777
1234567890123456789012345678901234567890123456789012345678901234567890123456789
 1
 2
 3
 4
 5
 6
 7
 8             NAME            <               >
 9
10             ADDRESS LINE 1  <                              >
11
12             ADDRESS LINE 2  <                              >
13
14             ADDRESS LINE 3  <                              >
15
16
17
18
19
20
21
22
23
24
25
```

Questions 4

1. Define a data item for the WORKING-STORAGE SECTION to define a date for input in the format dd mm yyyy. The dd, mm and yyyy data items must all be individually accessible by the program.

2. Define a data item named NAME-IN to hold a name with 40 alphanumeric characters.

3. Use the REDEFINES clause to redefine the data item NAME-IN so that it can also be accessed by first name 15 characters and surname 25 characters.