GAUTENG DEPARTMENT OF EDUCATION

SENIOR CERTIFICATE EXAMINATION

COMPUTER STUDIES HG
(First Paper: Practical)

TIME : 3 hours

MARKS : 100

INSTRUCTIONS:

- This paper consists of TWO sections. Only answer the section applicable to your programming language.
- If you studied **Delphi**, you must answer **Section A**. If you studied **Pascal**, you must answer **Section B**.
- No components may be added to or removed from the given Delphi forms, unless otherwise stated.
- Poor programming techniques will be penalised.
- The help function (F1) of your programming language may be consulted anytime during this examination.
- Save your work on a regular basis.
- Your complete examination number must appear on every page that you hand in.

SECTION A
DELPHI
QUESTION 1

Batch processing is a kind of processing where a number of transactions are initially written to a transaction file and later used to update a master file.

A clothing store wants you to write them a program that will handle the accounts of their clients. You must use TWO different files in the program: a master file and a transaction file. A data file must be used as the master file and a text file must be used as the transaction file.

After every transaction has been done, it must be written to the transaction file (text file). At the end of *each* day the master file (data file) must be updated by using the transactions that were written to the transaction file during that day.

A data file named **clientsD.dat**, is already stored on your examination disk and contains the information of the clients of the store. Immediately make a backup copy of this file on your stiffy or hard disk.

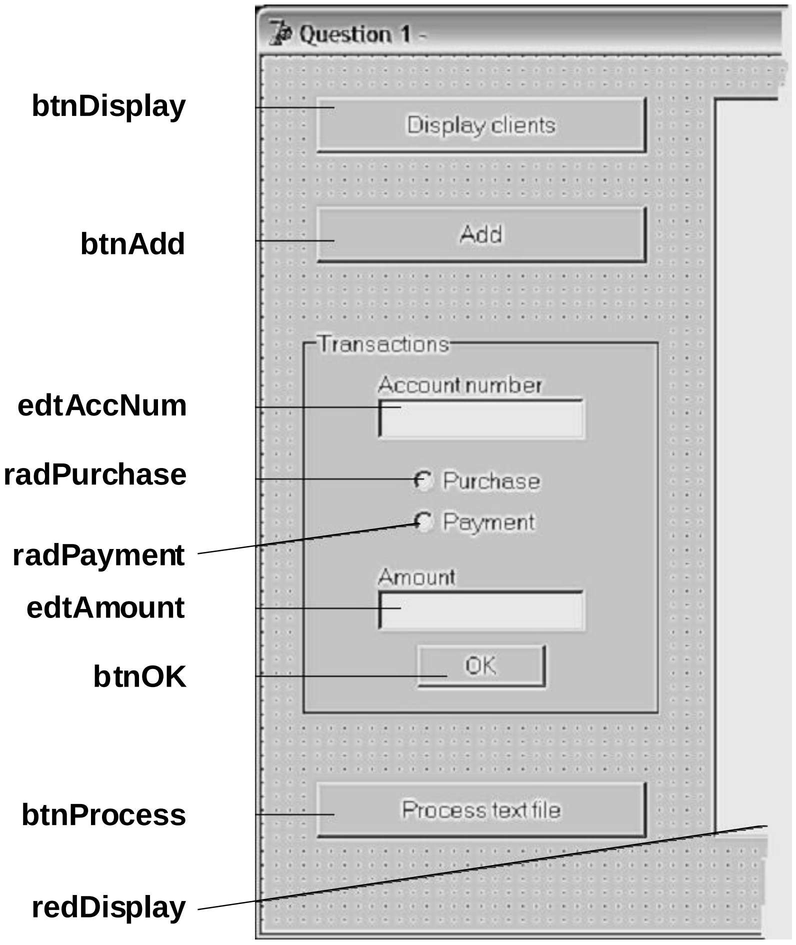
P.T.O.

Each record in the data file contains the following fields:

Account number	à	6 characters
Surname	à	30 characters
Initials	à	4 characters
Balance	à	real value

Open file '**`Q1P.dpr`**' in Delphi, go to File|Save As... and save the *unit* as '**`Q1U_XXXX.pas`**' (**`XXXX`** represents the last four digits of your examination number). Now go to File|Save Project As... and save the *project* as '**`Q1P_XXXX.dpr`**'.

1.1 Change **only** the *Caption* and *Name* properties of the different components on the form so that each corresponds with the figure below. Also add your examination number to the *Caption* of the form next to "Question 1". (2)



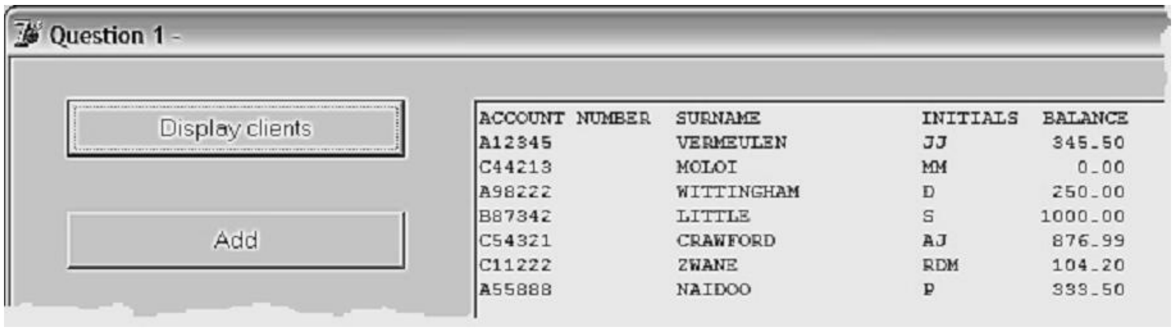
For each of the following questions, write an appropriate *Event Handler*:

1.2 **DISPLAY CONTENTS OF DATA FILE ON RICHEDIT**

Whenever the user clicks on **btnDisplay**, the contents of the master file (*clientsD.dat*) must be displayed on **redDisplay** in a columnar fashion.

- Each column must have an appropriate heading.
- Balances must be justified to the right. (13)

After the user has clicked on **btnDisplay**, the form should display as follows:



```
ACCOUNT NUMBER  SURNAME       INITIALS  BALANCE
A12345          VERMEULEN     JJ         345.50
C44213          MOLOI         MM           0.00
A98222          WITTINGHAM    D          250.00
B87342          LITTLE        S         1000.00
C54321          CRAWFORD      AJ         876.99
C11222          ZWANE         RDM        104.20
A55888          NAIDOO        P          333.50
```

1.3 **ADD NEW CLIENT TO THE END OF DATA FILE**

Whenever the user clicks on **btnAdd**, a *new* client's data must be added as follows to the end of the master file (*clientsD.dat*):

- The *account number*, *surname* and *initials* must be entered using *InputBoxes*.
- All new clients start with an opening balance of zero.

Using **btnAdd,** add the data of the following client:

- **MB STEYN** with account number of **B9865**

Check to see if the data was correctly added by clicking on **btnDisplay**. (7)

1.4 WRITE TRANSACTION TO TEXT FILE

Whenever the user clicks on **btnOK**, the user's input done on the Transactions panel, must be added to the end of a transaction file (***transac.txt***) in one line:

- The program must be able to either create the text file or, if it already exists, be able to add lines to the text file.
- The layout of the lines in the text file must be as follows: an account number followed by a comma, followed by a single character indicating the type of transaction and lastly the amount of the transaction.

The type of transaction can either be a purchase or a payment. Purchases are debit transactions on the client's account and payments are credit transactions on the client’s account.

D = Debit transaction (Client's balance will increase)
C = Credit transaction (Client's balance will decrease)

An excerpt from the transaction file (text file) may look as follows:

```
A11367,D510.40
C8698,C200.00
V342,D99.95
J98009,D748.99
P8708,C500.00
.
.
.
```

By using **btnOK**, add the following transactions:

- A12345 buys clothes to the value of R154.50
- C11222 makes a payment of R100.00 to his account

Check whether the text file (*transac.txt*) was created correctly by opening it in any text editor. The text file should display as follows: (10)

```
A12345,D154.50
C11222,C100.00
```

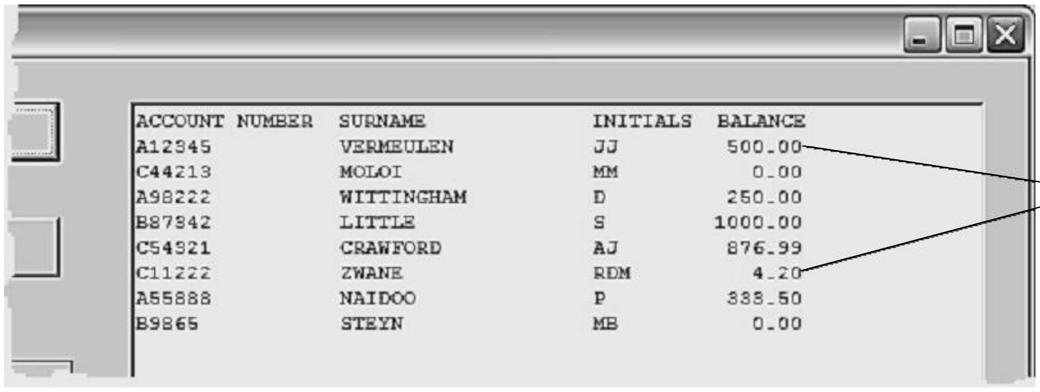
1.5 UPDATING MASTER FILE USING TRANSACTION FILE

NOTE: In case your program could not create the text file (*transac.txt*) in Question 1.4, you may create it yourself using any text editor.

Whenever the user clicks on **btnProcess**, the following must happen:

- Every transaction must be read separately from *transac.txt*.
- The client of that transaction must then be found in the master file (*clientsD.dat*). If the client was not found in the data file the transaction in the text file is regarded as invalid. An appropriate message must be displayed and the next transaction must be processed.
- The new balance of the client must be calculated according to the type of the transaction (D or C).
- The new balance must then be written back over the same record of the client in *clientsD.dat*.
- After all transactions in the transaction file (transac.txt) have been processed,
  - the transaction file (transac.txt) must be cleared from the disk, including all invalid transactions, and
  - an appropriate message must be displayed. (29)

If the update was successful, **btnDisplay** should display the following:



```
ACCOUNT NUMBER  SURNAME          INITIALS  BALANCE
A12345          VERMEULEN        JJ         500.00
C44213          MOLOI            MM           0.00
A98222          WITTINGHAM       D          250.00
B87342          LITTLE           S         1000.00
C54321          CRAWFORD         AJ         876.99
C11222          ZWANE            RDM          4.20
A55888          NAIDOO           P          333.50
B9865           STEYN            MB           0.00
```

The following must be handed in for Question 1 (Delphi):
- A Printout of **`Q1U_XXXX.pas`**.
- An "Alt | Print Scrn" of the form *while the output of **Question 1.5** is being displayed*.

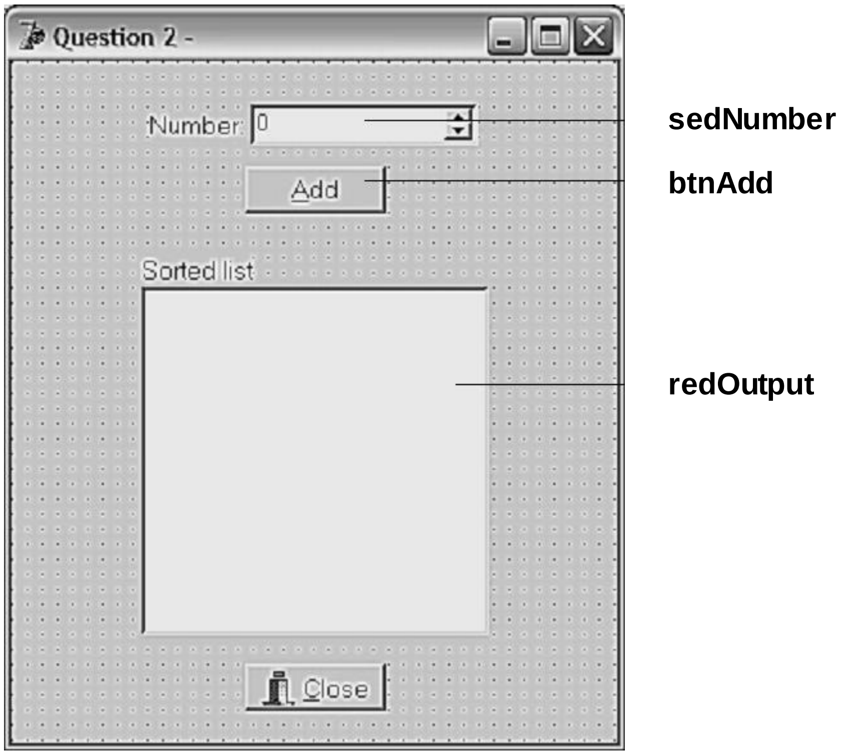
[61]

QUESTION 2

The owners of a stall at a fête must keep track of the number of marshmallows a person can eat in one minute. They need a program that will ask for each participant's number of marshmallows eaten in one minute and then display a sorted list of the numbers. The list of numbers must always be sorted from the highest to the lowest number of marshmallows.

Open the file **`'Ques2P.dpr'`** in Delphi, go to File|Save As... and save the *unit* as '**`Ques2U_XXXX.pas`**' (**`XXXX`** represents the last four digits of your examination number). Now go to File|Save Project As... and save the *project* as '**`Ques2P_XXXX.dpr`**'.

2.1 Change **only** the *Caption* and *Name* properties of the components on the form so that each corresponds with the figure below. Also add your examination number to the *Caption* of the form next to "Question 2".

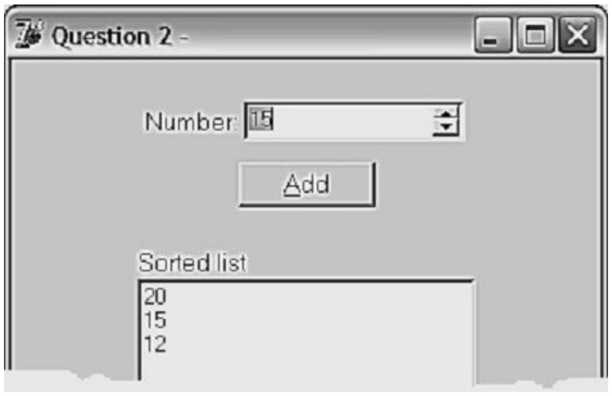


The form above will be used to always show the latest list of numbers. Only the TEN best numbers must be displayed.

2.2 Every time a new number of marshmallows eaten has been added using **btnAdd**, the number must be displayed in its correct position on **redOutput** so that the list will always be sorted from the highest number to the lowest number.

Tip: Store the new number at position 11 in an array, then sort the entire array again.

After the THREE numbers, 12, 20 and 15 have been added, the output must display as follows:



```
20
15
12
```

The program must adhere to the following:

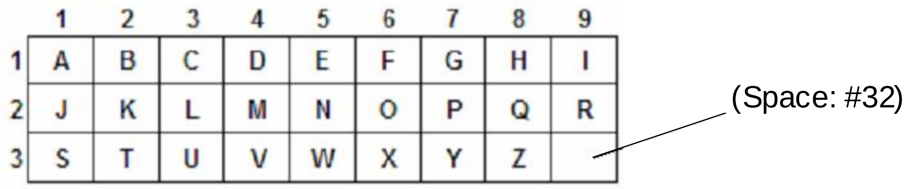
- A global array must be used to store the numbers.
- Write a procedure, **Swap**, that uses only TWO parameters for the swapping of TWO numbers. This procedure must be called during the sorting of the array.
- You need only write ONE procedure (for the swapping of the two numbers).

The following must be handed in for Question 2 (Delphi):
- A printout of **`Ques2U_XXXX.pas.`**

[23]

QUESTION 3

To decipher a secret message, the letters of the alphabet must be placed in a matrix as follows:



	1	2	3	4	5	6	7	8	9
1	A	B	C	D	E	F	G	H	I
2	J	K	L	M	N	O	P	Q	R
3	S	T	U	V	W	X	Y	Z	

(Space: #32)

The character in **row 3, column 9** must be a **space** (ASCII #32).

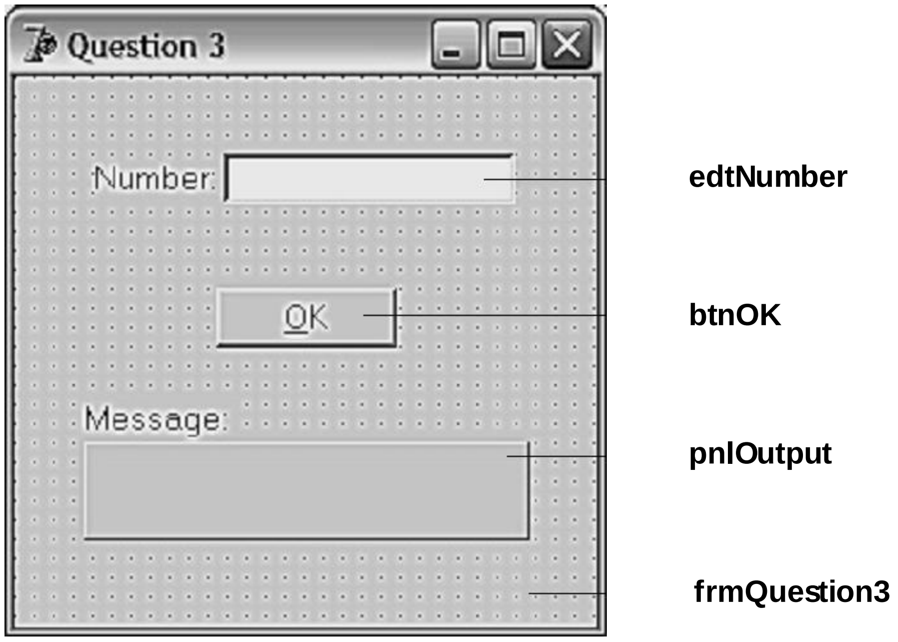
The message will always consist of an unknown number of 2-digit numbers. The first digit of each number represents the row on the matrix and the second represents the column.

Example: the number **27** represents the letter in row **2**, column **7**, which is a '**P**'.

A full message may thus look as follows:
Message: 19 32 39 19 31 39 14 26 25 15
Meaning: I T I S D O N E

Start with a new application in Delphi and save the *unit* as **`XXXX_Q3U.pas`** and the *project* as **`XXXX_Q3P.dpr`** (**XXXX** represents the last four digits of your examination number).

3.1 Design a form like the one below and add your examination number to the *Caption* of the form next to "Question 3". (2)



P.T.O.

3.2 As soon as the program is run, the letters of the alphabet, as well as a space, must be placed in a matrix the way it was explained before. (7)

3.3 Whenever the user clicks on **btnOK**, the following must happen:

- The number that was entered into **edtNumber**, must be used to extract the correct letter from the matrix.
- This new letter must each time be added to the letters already on **pnlOutput** and again be displayed on **pnlOutput**. (7)

Test your program by deciphering the following message:
24 37 39 27 29 26 17 29 11 24 39 35 26 29 22 31

The following must be handed in for Question 3 (Delphi):
- A Printout of **`XXXX_Q3U.pas`**.
- An "Alt | Print Scrn" of the form *while the deciphered message from **Question 3.3** is being displayed*.

[16]

TOTAL FOR SECTION A: [100]

SECTION B
PASCAL
QUESTION 4

Batch processing is a kind of processing where a number of transactions are initially written to a transaction file and later used to update a master file.

A clothing store wants you to write them a program that will handle the accounts of their clients. You must use TWO different files in the program: a master file and a transaction file. A data file must be used as the master file and a text file must be used as the transaction file.

After every transaction has been done, it must be written to the transaction file (text file). At the end of *each* day the master file (data file) must be updated by using the transactions that were written to the transaction file during that day.

A data file named **clientsP.dat**, is already stored on your examination disk and contains the information of the clients of the store. Immediately make a backup copy of this file on your stiffy or hard disk.

Each record in the data file contains the following fields:

Account number	à	6 characters
Surname	à	30 characters
Initials	à	4 characters
Balance	à	real value

Start with a new program in Turbo Pascal and save it as '**`Q4_XXXX.pas`**' (**`XXXX`** represents the last four digits of your examination number). The program must adhere to the following:

4.1 The program must **repeatedly** display the menu below until the user chooses **Option 5**: (2)

```
1. Display clients.
2. Add new client.
3. Do transaction.
4. Process transaction file.
5. Quit.
```

For each of the following questions, write a procedure **without any parameters**.

4.2 **DISPLAY CONTENTS OF DATA FILE ON SCREEN**

Write a procedure, **Display**, for **Option 1** that will display the contents of the master file (*clientsP.dat*) on screen in a columnar fashion.

- Each column must have an appropriate heading.
- Balances must be justified to the right. (13)

After the user has chosen **Option 1**, the output should display as follows:

```
ACCOUNT NUMBER   SURNAME        INITIALS       BALANCE
A12345           VERMEULEN      JJ              345.50
C44213           MOLOI          MM                0.00
A98222           WITTINGHAM     D               250.00
B87342           LITTLE         S              1000.00
C54321           CRAWFORD       AJ              876.99
C11222           ZWANE          RDM             104.20
A55888           NAIDOO         P               333.50
```

4.3 **ADD NEW CLIENT TO THE END OF DATA FILE**

Write a procedure, **NewClient**, for **Option 2** that will add a *new* client's data as follows to the end of the master file (*clientsP.dat*):

- The user must be prompted for the *account number*, *surname and initials*.
- All new clients start with an opening balance of zero. (7)

By using **Option 2**, add the data of the following client:

- **MB STEYN** with account number **B9865**

Check to see if the data was added correctly by choosing **Option 1** again.

4.4 WRITE TRANSACTION TO TEXT FILE

Write a procedure, **Add**, for **Option 3** that will prompt the user to type in the *account number, type of transaction* and *amount* as three separate inputs. This input must then be added to the end of a transaction file (***transac.txt***) in one line.

- The program must be able to either create the text file or, if it already exists, be able to append to the text file.
- The layout of the lines in the text file must be as follows: an account number followed by a comma, followed by a single character indicating the type of transaction and lastly the amount of the transaction.

The type of transaction can either be a purchase or a payment. Purchases are debit transactions on the client's account and payments are credit transactions on the client's account.

D = Debit transaction (Client's balance will increase)
C = Credit transaction (Client's balance will decrease)

An excerpt from the transaction file (text file) may look as follows:

```
A11367,D510.40
C86982,C200.00
V34256,D99.95
J98009,D748.99
P8708,C500.00
.
.
.
```

Using **Option 3**, add the following transactions:

- A12345 buys clothes to the value of R154.50
- C11222 makes a payment of R100.00 in favour of his account

Check whether the text file (*transac.txt*) was created correctly by opening it in any text editor. The text file should display as follows: (10)

```
A12345,D154.50
C11222,C100.00
```

4.5 UPDATING MASTER FILE USING TRANSACTION FILE

NOTE: In case your program could not create the text file (*transac.txt*) in Question 4.4, you may create it yourself using any text editor.

Write a procedure, **Process**, for **Option 4** that will do the following:

- Every transaction must be read separately from *transac.txt*.
- The client of that transaction must then be found in the master file (*clientsP.dat*). If the client was not found in the data file the transaction in the text file is regarded as invalid. An appropriate message must be displayed and the next transaction must be processed.
- The new balance of the client must be calculated according to the type of the transaction (D or C).
- The new balance must then be written back to the same record of the client in *clientsP.dat*.
- After all transactions in the transaction file (transac.txt) have been processed,
  – the transaction file must be cleared from the disk, including all invalid transactions, and
  – an appropriate message must be displayed. (29)

If the update was successful, **Option 1** should display the following:

```
ACCOUNT NUMBER  SURNAME          INITIALS      BALANCE
A12345          VERMEULEN        JJ             500.00
C44213          MOLOI            MM               0.00
A98222          WITTINGHAM       D              250.00
B87342          LITTLE           S             1000.00
C54321          CRAWFORD         AJ             876.99
C11222          ZWANE            RDM              4.20
A55888          NAIDOO           P              333.50
```

New balances

The following must be handed in for Question 4 (Pascal):
- A printout of **`Q4_XXXX.pas`**.

[61]

QUESTION 5

The owners of a stall at a fête must keep track of the number of marshmallows a person can eat in one minute. They need a program that will ask for each participant's number of marshmallows eaten in one minute and then display a sorted list of the numbers. The list of numbers must always be sorted from the highest to the lowest number of marshmallows.

Start with a new program in Turbo Pascal and save it as '**`XXXX_Q5.pas`**' (**`XXXX`** represents the last four digits of your examination number).

This program will be used to always show the latest list of numbers. Only the TEN best numbers must be displayed.

When the program is run, the latest list must be displayed at the top of the screen with a prompt for the next number of marshmallows just below the list. A new number must be asked until the user enters -1. Each number entered must appear in the list in its correct position so that the list will always be sorted from the highest number to the lowest number.

Tip: Store the new number at position 11 in an array, then sort the entire array again.

After the three numbers, 12, 20 and 15 have been added, the output must display as follows:

```
Sorted list
20
15
12

Enter the newest number.
_
```

The program must adhere to the following:

- Write a procedure, **Swap**, that uses only TWO parameters for the swapping of TWO numbers. This procedure must be called during the sorting of the array.
- You need only write ONE procedure (for the swapping of the two numbers).

The following must be handed in for Question 5 (Pascal):
- A printout of **`XXXX_Q5.pas`**.

[23]

QUESTION 6

To decipher a secret message, the letters of the alphabet must be placed in a matrix as follows:

	1	2	3	4	5	6	7	8	9
1	A	B	C	D	E	F	G	H	I
2	J	K	L	M	N	O	P	Q	R
3	S	T	U	V	W	X	Y	Z	

(Space: #32)

The character in **row 3, column 9** must be a **space** (ASCII #32).

The message will always consist of an unknown number of 2-digit numbers. The first digit of each number represents the row on the matrix and the second represents the column.

Example: the number **27** represents the letter in row **2**, column **7**, which is a '**P**'.

A full message may thus look as follows:
Message: 19 32 39 19 31 39 14 26 25 15
Meaning: I T I S D O N E

Start with a new program in Turbo Pascal and save it as '**`Q6_XXXX.pas`**' (**XXXX** represents the last four digits of your examination number).

6.1 As soon as the program is run, the letters of the alphabet, as well as a space, must be placed in a matrix the way it was explained above. (7)

6.2 The program must adhere to the following:

- The user must repeatedly be prompted to enter a number until the number –1 is entered, after which the program must stop.
- Each number the user enters must be used to extract the correct letter from the matrix.
- This new letter must each time be added to the letters already being displayed and again be displayed onscreen. (9)

Test your program by deciphering the following message:
24 37 39 27 29 26 17 29 11 24 39 35 26 29 22 31

The following must be handed in for Question 6 (Pascal):
- A printout of **`Q6_XXXX.pas`**.

[16]

TOTAL: 100

END