Set B Part 1

Design Marks: Total = 7

account_details(n, A, B)

// Array A and B of size *n* initialised into -1.

1. read the value of *m* //number of customers
2. initialise an array C of size *m* //to track the positions of customers in the order of *acc_no*
3. j = 0 //keep track of index in array C
4. **for** i = 1 **to** *m*

   do read *acc_no* //account number
   read *b* //balance
   compute *p = b mod n* //find position *p*

   do

   if A[*p*] = -1

   //assign *acc_no, b into* position *p* of array A and B respectively

   then A[*p*] = *acc_no*
   B[*p*] = *b*
   C[*j++*] = *p*

   else

   //find *k* which is the next vacant position

   for *pos* = 1 to *n*

   do

   compute *k = (p+pos) mod n*

   if A[k] = -1

   then

   A[*k*] = *acc_no*
   B[*k*] = *b*
   C[*j++*] = *k*
   break // break out of for

5. **for** i = 1 **to** *m*

   print C[*i*] //print position *p* of each customer in the order of *acc_no*

Evaluation criteria : **[6 marks]**
Division:

- Read account number and balance of a customer and store into the variables - 1 mark
- Find the position p of the customer in the arrays A and B - 1 mark
- If position p is vacant, then store the details of the customer in the arrays A and B at position p - 1 mark
- If position p is not vacant find next vacant position in the arrays A and B and store - 1 mark
- Store the position p in an array as per the order of the account number and print the positions after storing m customer details - 1 mark
- Correct function name, and number of arguments - 1 mark

display_balance(A,B,n)

```
1. for i = 1 to n
       do
           if A[i] = -1
                   print -1  //array vacant
           else
                   print A[i]  B[i] separated by a space
```

Evaluation criteria : **[1 mark]**
Division: Print account number and balance of each customer in the arrays separated by a space - 1 mark

Design Marks: Total = 3

Set B Part 2

account_details(n,m, A, B)

//A: 2D Array of size *n*m* initialised into -1 (to store account number)

//B: 2D Array of size *n*m* initialised into -1 (to store balance)

1. **for** j = 1 **to** *m*
   do
   read *acc_no* //account number
   read *b* //balance
   compute *p = b mod n*
   for i = 1 **to** n
   do
   if A[*p*][*i*] = -1
   then
   A[*p*][*i*] = *acc_no*
   B[*p*][*i*] = *b*
   break // break out of for

Evaluation criteria : **[1 mark]** Division: • Read the account number and balance of the customers and compute position p - 0.25 mark • Selection of proper data structure to store the details of customers (if more than one customer get same position p) - 0.75 mark

arrange_customers(n, m, A, B)

//A and B are 2-dimensional array of size *n*m* with account number details and balance details respectively

1. **for** *row* = 1 to *n*
   do
   if B[*row*][1] != -1
   do
   //apply any sorting algorithm on B[*row*], while swapping elements in B[*row*] make changes accordingly in A[*row*]

```
//example given below uses Bubble sort algorithm
for i = 1 to m
    for  j = 1 to m-i
        do
            if B[row][j] > B[row][j+1]
                then
                    swap B[row][j] , B[row][j+1]
                    swap A[row][j] , A[row][j+1]
```

Evaluation criteria : **[1 mark]** Division: • Selection of a sorting algorithm - 0.25 mark • Sort array B (non-decreasing order of balances at position p), and reflect the same changes in array A - 0.75 mark

display_balance(n,m,A,B)

//each new line prints account number and balance b (separated by a comma) of the customers that are allotted to position p, where each customer is separated by a space

```
1. for  i =1 to n
        j = 0
        do
            if A[i][j] = -1
                then
                    print('NULL')
            else
                while A[i][j] != -1
                    print A[i][j],B[i][j] print(' ')    //prints all the customers
                allotted to same position separated by space
                    j++      //increment j by one
```

Evaluation criteria : **[1 mark]** Division: Print the details of the customers in the given format - **1** mark