執行環境：Windows

程式語言：C

程式內容說明：

```
typedef struct matrixnode *MatrixPointer;      //refer to textbook
typedef struct
{
        int row;
        int col;
        int value;
}matrixentry;
struct matrixnode
{
        MatrixPointer down;
        MatrixPointer right;
        union{
                matrixentry entry;
                MatrixPointer next;
        }u;
};
```

MatrixPointer MatrixInput(...) =用來輸入資料

void MatrixPrint(MatrixPointer M,MatrixPointer *headnode)=打印出矩陣

void MatrixFree(MatrixPointer M,MatrixPointer *headnode)=釋放空間回系統

主要執行區塊〈矩陣相乘〉:

MatrixPointer multiplication(...)

```
MatrixPointer multiplication(MatrixPointer A,MatrixPointer B,MatrixPointer *headnode,int headnodenum)
{
		int total = 0,i,cellvalue = 0;
		MatrixPointer head,temp,last,headA,headB,entryA,entryB;

		head = malloc(sizeof(*head));			//create head, recording matrix info., for result of multiplication
		head->u.entry.row = A->u.entry.row;		//dim. of result take row from A and column from B
		head->u.entry.col = B->u.entry.col;

		for(i = 0 ; i < headnodenum ; i++)		//construct the headnodes
		{
			temp = malloc(sizeof(*temp));
			headnode[i] = temp;
			headnode[i]->right = temp;
			headnode[i]->u.next = temp;
		}
		head->right = headnode[0];

		headA = A->right;			//enter first headnode in headnodes of A
	headA = A->right;			//enter first headnode in headnodes of A
	for(i = 0 ; i < A->u.entry.row ; i++)
	{
		last = headnode[i];		//last is a temporary space to link entries into a row
		entryA = headA->right;	//enter first entry in current row
		entryB = B->right->down; //enter first entry in first column
		for(headB = B->right ; headB != B ;)
		{
			if(entryA == headA || entryB == headB) //scanned over the current row, go back to first entry
			{				//or scanned over the current column, go to the entry of next column
				if(cellvalue)	//if having value, store it( similar to the part in MatrixInput)
				{
					temp = malloc(sizeof(*temp));		//construct the entry node
					temp->u.entry.row = i;
					temp->u.entry.col = headB->down->u.entry.col;
					temp->u.entry.value = cellvalue;
					last->right = temp;
					last = temp;
					headnode[headB->down->u.entry.col]->u.next->down = temp;  //link entry into column
					headnode[headB->down->u.entry.col]->u.next = temp;
						//next of headnode is also a temporary space to link entries into column
					total++;
				}
				cellvalue = 0;
				entryA = headA->right;		//back to first entry
				headB = headB->u.next;		//go to next column
				entryB = headB->down;		//first entry of next column
			}
```

```
                    }
                    else switch((entryA->u.entry.col == entryB->u.entry.row)? 0:(entryA->u.entry.col < entryB->u.entry.row)? -1:1)
                    {    //compare column of A and row of B
                         case -1:  //row of B is ahead of column of A
                              entryA = entryA->right;
                              break;
                         case 0:   //do multiplication and store result
                              cellvalue = cellvalue + entryA->u.entry.value*entryB->u.entry.value;
                              entryA = entryA->right;
                              entryB = entryB->down;
                              break;
                         case 1:   //column of A is ahead of row of B
                              entryB = entryB->down;
                    }
               }
               last->right = headnode[i]; //close row list
               headA = headA->u.next;     //go to next row of A
          }
               head->u.entry.value = total;

               for(i = 0 ; i < headnodenum ; i++) headnode[i]->u.next->down = headnode[i]; //close column list
               for(i = 0 ; i < headnodenum-1 ; i++) headnode[i]->u.next = headnode[i+1];   //link head nodes
               headnode[headnodenum-1]->u.next = head; //last headnode requires to point to head

               return head;
}
```