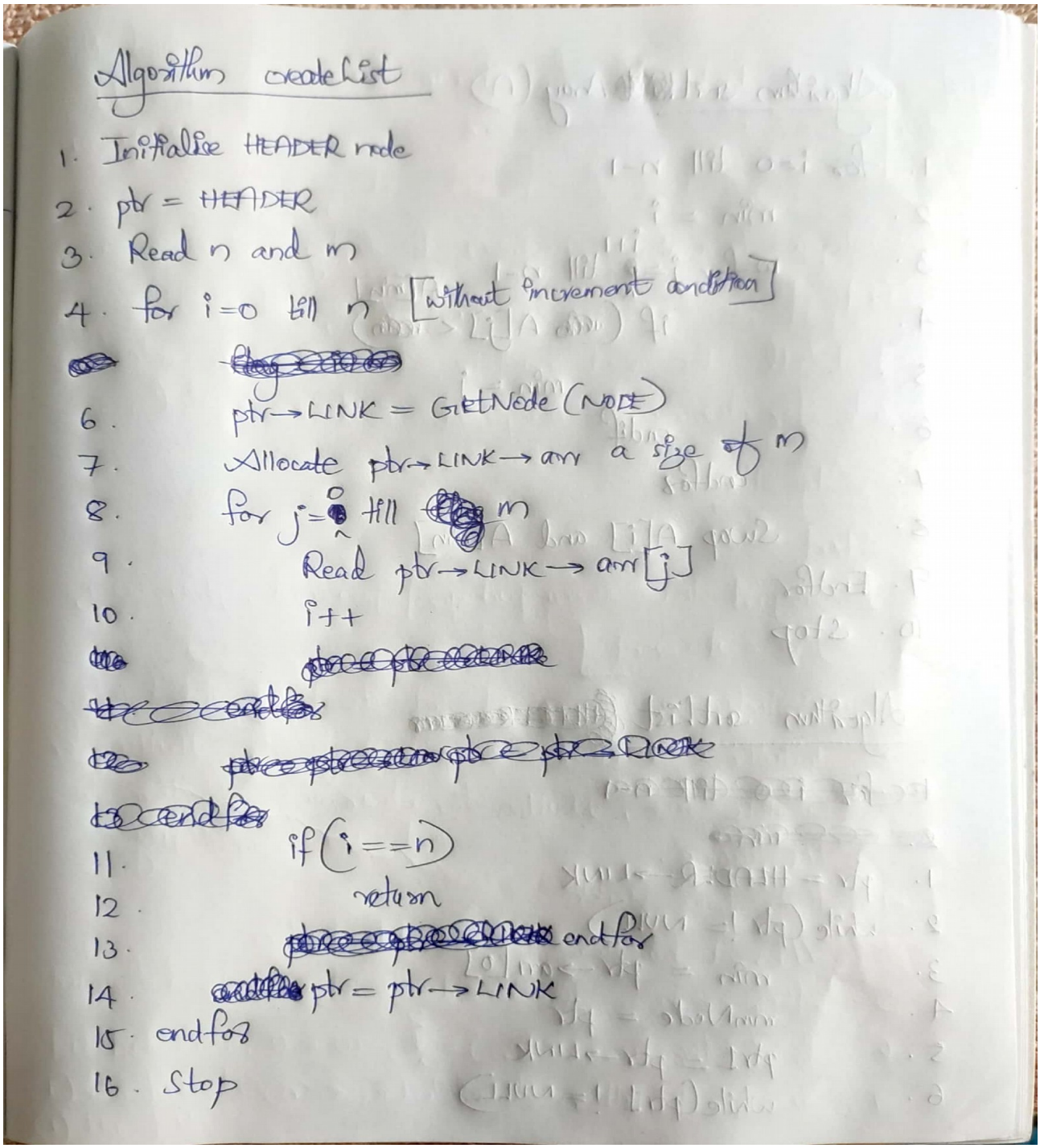
DATA STRUCTURES INTERNAL EXAMINATION

AMAL NATH M

R3 11

ALGORITHM -



Algorithm sortArray (A)

1. for i=0 till n-1
2. min = i
3. for j=i+1 till n A[min]
4. if (A[j] <)
5. min = j
6. endif
7. endfor
8. Swap A[i] and A[min]
9. Endfor
10. stop

Algorithm sortList

1. ptr = HEADER→LINK
2. while (ptr != NULL)
3. min = ptr→arr[0]
4. minNode = ptr
5. ptr1 = ptr→LINK
6. while(ptr1 != NULL)

```
7.      if (ptr1->arr[0] < min)
8.          min = ptr1->arr[0]
9.          minNode = ptr1
10.     endif
        ptr1 = ptr1->LINK
11.     Interchange minNode and ptr
12.   endwhile
13.   Interchange minNode and ptr
14.   ptr = ptr->LINK
15.  Endwhile
16. Stop
```

Algorithm display

```
1. ptr = HEADER->LINK
2. for (i=0) till n  [without increment]
4.     for j=0 till m
5.         Print ptr->arr[j]
6.         i++
7.         if (i==n)
8.             return
9.         endif
10.    endfor
11.    ptr = ptr->LINK
12. endfor
```

PROGRAM CODE -

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
	int* arr;
	struct node* LINK;
};

int m, n;

void createList(struct node* ptr)
{
	printf("Enter the value of n\n");
	scanf("%d", &n);
	printf("Enter the value of m\n");
	scanf("%d", &m);

	printf("Enter the numbers\n");

	for(int i=0; ; )
	{
		ptr->LINK = (struct node*) malloc(sizeof(struct node));
		ptr->LINK->arr = malloc(m*sizeof(int));

		for(int j=0; j<m; j++)
		{
			scanf("%d", &ptr->LINK->arr[j]);
			i++;

			if(i == n)
				return;
		}
		ptr = ptr->LINK;
	}
}

void sortArray(int* A, int m)
{
	for(int i=0; i < m-1; i++)
	{
		int min = i;

		for(int j=i+1; j < m; j++)
			if(A[j] < A[min])
				min = j;

		int temp = A[i];
		A[i] = A[min];
		A[min] = temp;
```

```
	}
}

//INTERCHANGING LINKS
void sort(struct node* ptrP)
{
	struct node* ptr = ptrP->LINK;

	while(ptr != NULL)
	{
		int min = ptr->arr[0];
		struct node* minP = ptrP;
		struct node* minNode = ptr;

		struct node* ptr1P = ptr;
		struct node* ptr1 = ptr->LINK;

		while(ptr1 != NULL)
		{
			if(ptr1->arr[0] < min)
			{
				min = ptr1->arr[0];
				minP = ptr1P;
				minNode = ptr1;
			}
			ptr1P = ptr1;
			ptr1 = ptr1->LINK;
		}

		struct node* temp = ptrP->LINK;
		ptrP->LINK = minP->LINK;
		minP->LINK = temp;

		temp = ptr->LINK;
		ptr->LINK = minNode->LINK;
		minNode->LINK = temp;

		ptrP = ptr;
		ptr = ptr->LINK;
	}
}

//SWAPPING DATA
void sortList(struct node* ptr)
{
	while(ptr != NULL)
	{
		int min = ptr->arr[0];
		struct node* minNode = ptr;
		struct node* ptr1 = ptr->LINK;
		int *temp;
```

```
                while(ptr1 != NULL)
                {
                        if(ptr1->arr[0] < min)
                        {
                                min = ptr1->arr[0];
                                minNode = ptr1;
                        }
                        ptr1 = ptr1->LINK;
                }

                temp = minNode->arr;
                minNode->arr = ptr->arr;
                ptr->arr = temp;

                ptr = ptr->LINK;
        }
}

void display(struct node* ptr)
{
        printf("The list: \n");
        for(int i=0; ; )
        {
                for(int j=0; j<m; j++)
                {
                        if(ptr->arr[j] != 0)
                        {
                                printf("%d ", ptr->arr[j]);
                                i++;

                                if(i == n)
                                        return;
                        }
                }

                printf("\n");

                ptr = ptr->LINK;
        }
}

void main()
{
        struct node* HEADER = (struct node*) malloc(sizeof(struct node));

        createList(HEADER);
        display(HEADER->LINK);

        struct node* ptr = HEADER->LINK;

        int flag = m;
```

```
        while(ptr != NULL)
        {
                if(ptr->LINK == NULL)
                        flag = n % m;
                sortArray(ptr->arr, flag);
                ptr = ptr->LINK;
        }

        printf("\n\nAfter array sort\n");
        display(HEADER->LINK);

        sortList(HEADER->LINK);
        //sort(HEADER);

        printf("\n\nAfter list sort\n");
        display(HEADER->LINK);
        printf("\n");
}
```

SAMPLE OUTPUT -

```
Enter the value of n
8
Enter the value of m
3
Enter the numbers
3
8
4
5
13
1
42
21
The list:
3 8 4
5 13 1
42 21

After array sort
The list:
3 4 8
1 5 13
21 42

After list sort
The list:
1 5 13
3 4 8
21 42
```