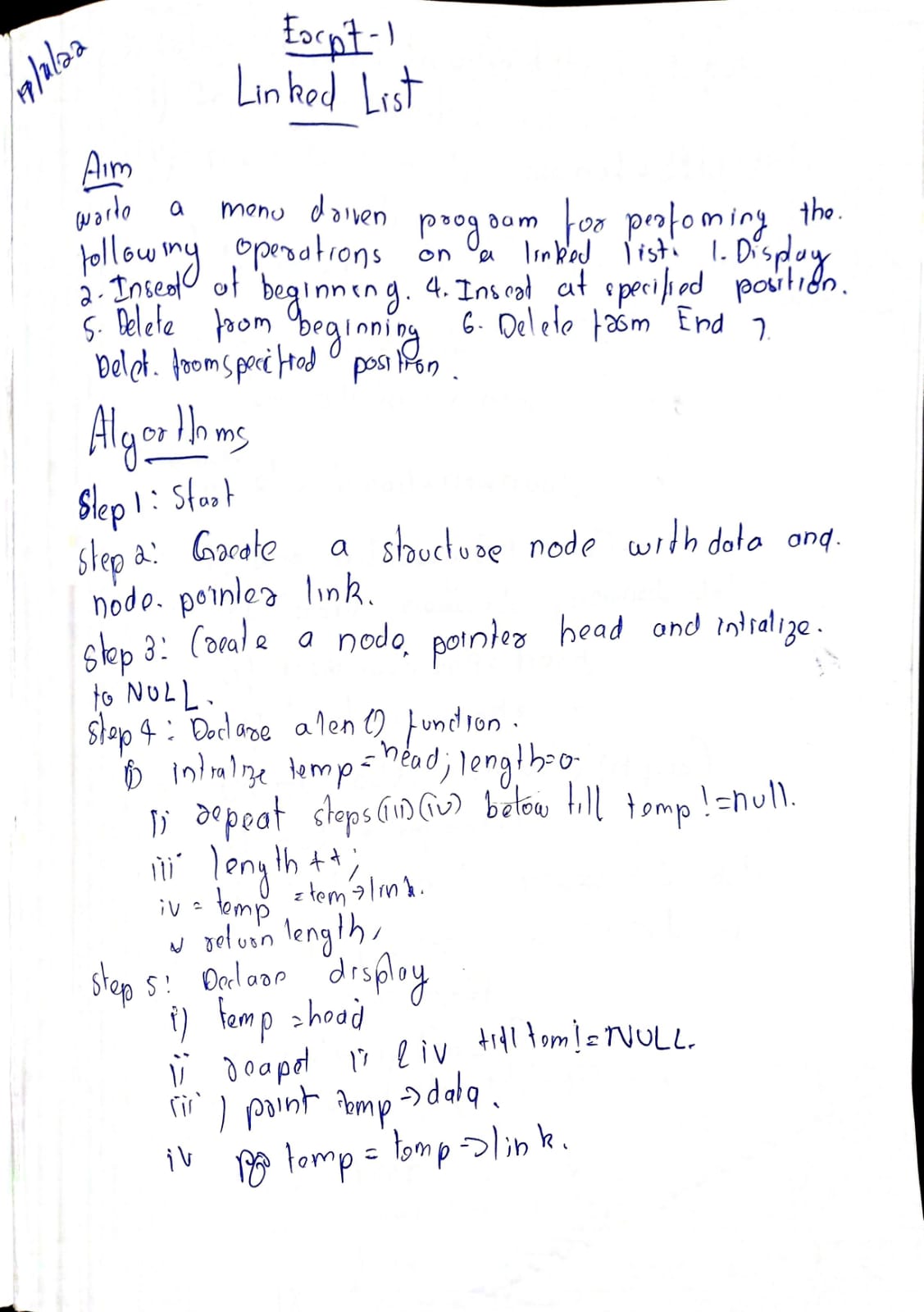
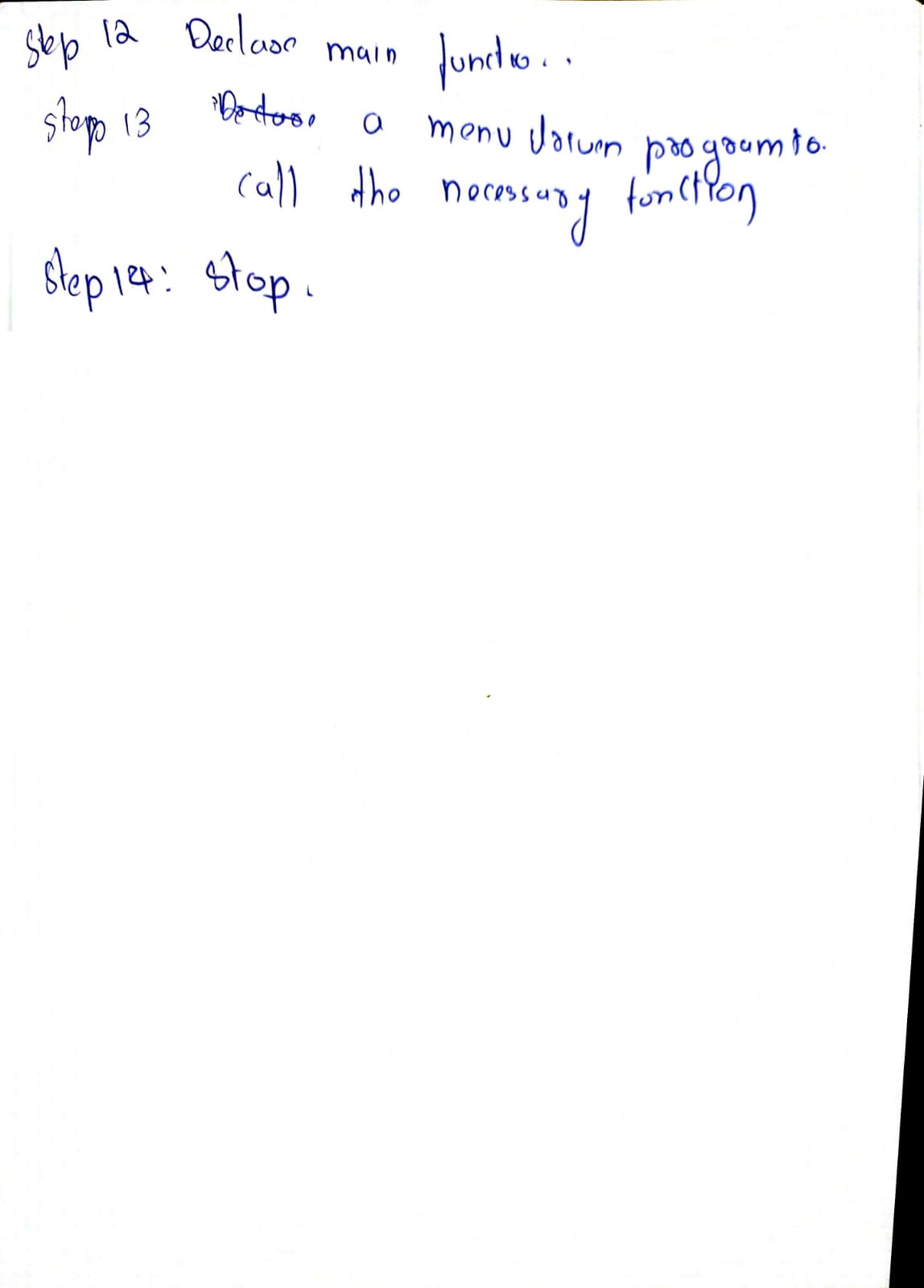
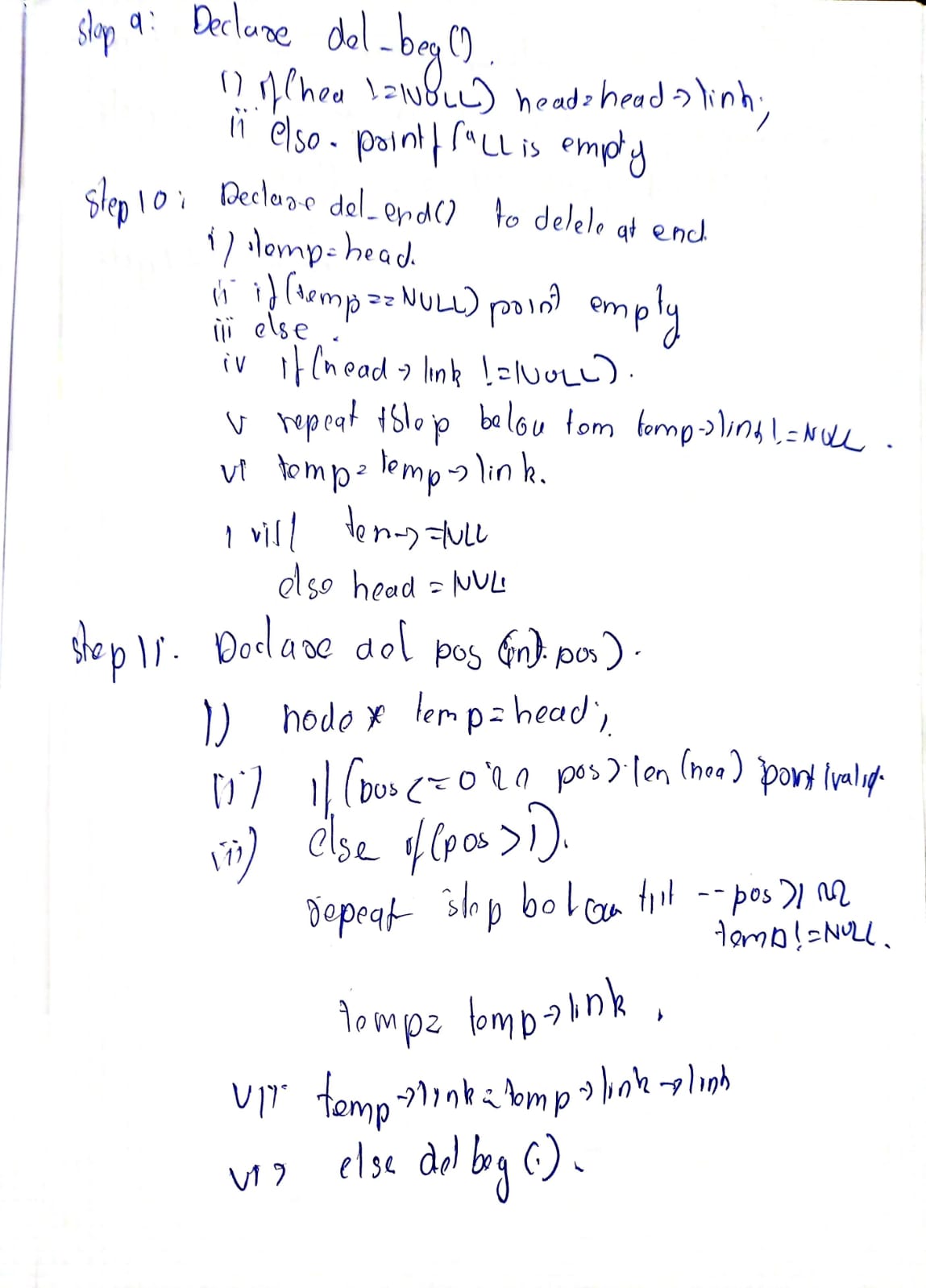
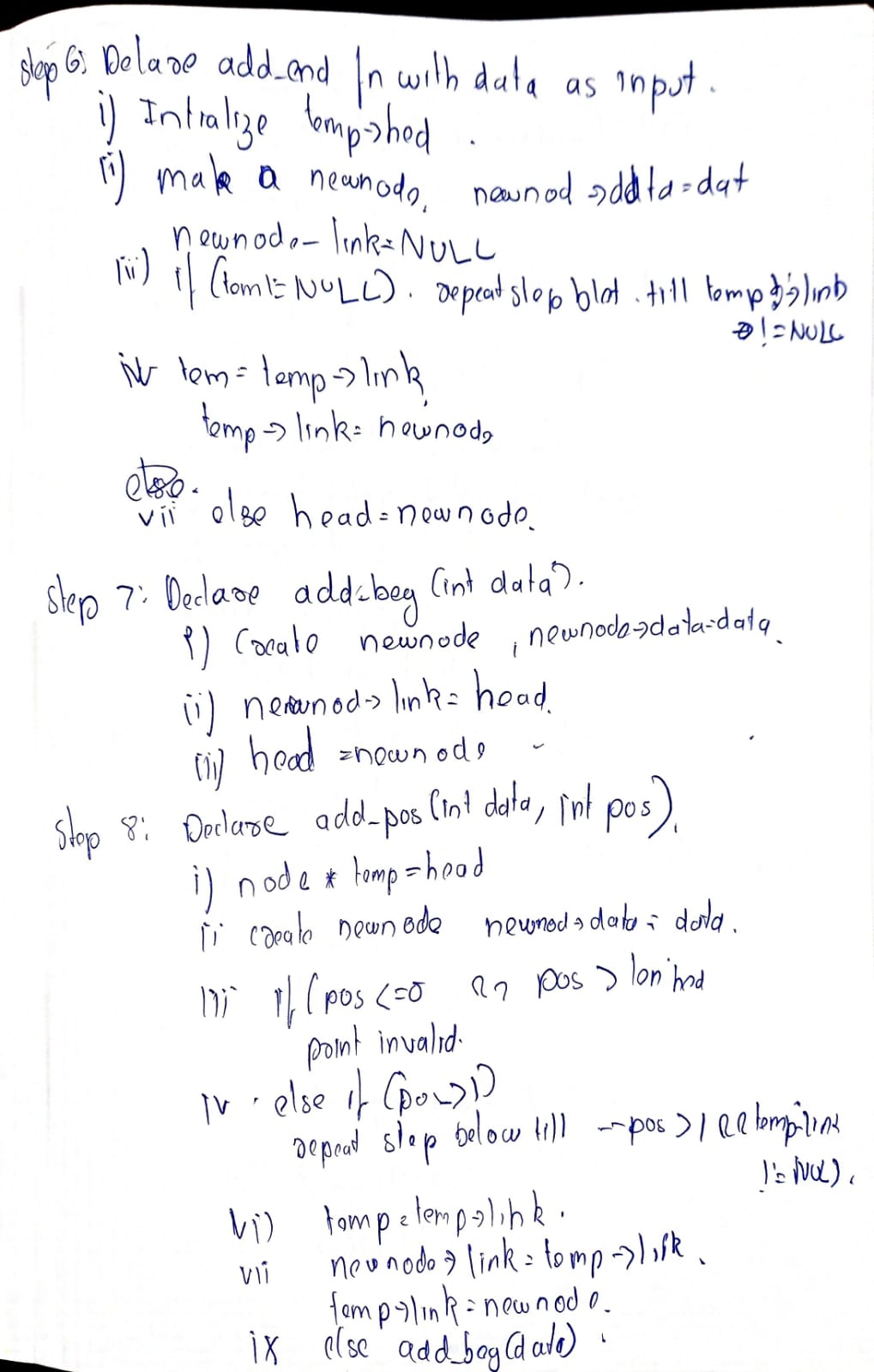
1.



**Code**

#include <stdio.h>

#include <stdlib.h>

*struct* node

{

*int* data;

*struct* node \*link;

};

*struct* node \*head = NULL;

*int* length()

{

*struct* node \*temp = head;

*int* length = 0;

    while (temp != NULL)

    {

        temp = temp->link;

        length++;

    }

    return length;

}

*void* display()

{

*struct* node \*temp = head;

    while (temp != NULL)

    {

        printf("%d ", temp->data);

        temp = temp->link;

    }

    printf("\n");

}

*void* add\_end(*int* *data*)

{

*struct* node \*temp = head;

*struct* node \*newnode = (*struct* node \*)malloc(sizeof(*struct* node));

    newnode->data = *data*;

    newnode->link = NULL;

    if (temp != NULL)

    {

        while (temp->link != NULL)

        {

            temp = temp->link;

        }

        temp->link = newnode;

    }

    else

    {

        head = newnode;

    }

}

*void* add\_beg(*int* *data*)

{

*struct* node \*newnode = (*struct* node \*)malloc(sizeof(*struct* node));

    newnode->data = *data*;

    newnode->link = head;

    head = newnode;

}

*void* add\_pos(*int* *data*, *int* *pos*)

{

*struct* node \*temp = head;

*struct* node \*newnode = (*struct* node \*)malloc(sizeof(*struct* node));

    newnode->data = *data*;

    if (*pos* > length() || *pos* < 1)

    {

        printf("Invalid Position\n");

    }

    if (*pos* > 1)

    {

        while (--*pos* > 1 && temp != NULL)

        {

            temp = temp->link;

        }

        newnode->link = temp->link;

        temp->link = newnode;

    }

    else

    {

        add\_beg(*data*);

    }

}

*void* del\_beg()

{

    if (head != NULL)

    {

        head = head->link;

    }

    else

    {

        printf("Linkd List is empty\n");

    }

}

*void* del\_end()

{

*struct* node \*temp = head;

    if (temp == NULL)

    {

        printf("The Linked List is empty\n");

    }

    else

    {

        if (head->link != NULL)

        {

            while (temp->link->link != NULL)

            {

                temp = temp->link;

            }

            temp->link = NULL;

        }

        else

        {

            head = NULL;

        }

    }

}

*void* del\_pos(*int* *pos*)

{

*struct* node \*temp = head;

    // if(pos>length() || pos<1)

    // {

    //  printf("Invalid Position\n");

    // }

    if (*pos* > 1)

    {

        while (--*pos* > 1 && temp != NULL)

        {

            temp = temp->link;

        }

        temp->link = temp->link->link;

    }

    else

    {

        del\_beg();

    }

}

*int* main()

{

    while (1)

    {

*int* choice;

        printf("1.Display\n2.Insert at End\n3.Insert at Beginning\n4.Delete from beginning\n5.Delete from end\n6.Insert into position\n7.Delete from position\n8.Exit");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

        {

            display();

            break;

        }

        case 2:

        {

*int* temp;

            printf("Enter the data");

            scanf("%d", &temp);

            add\_end(temp);

            display();

            break;

        }

        case 3:

        {

*int* temp;

            printf("Enter the data");

            scanf("%d", &temp);

            add\_beg(temp);

            display();

            break;

        }

        case 6:

        {

*int* temp, pos;

            printf("Enter the data and position");

            scanf("%d %d", &temp, &pos);

            add\_pos(temp, pos);

            display();

            break;

        }

        case 4:

        {

            del\_beg();

            display();

            break;

        }

        case 5:

        {

            del\_end();

            display();

            break;

        }

        case 7:

        {

*int* pos;

            printf("Enter the position");

            scanf("%d", &pos);

            del\_pos(pos);

            display();

            break;

        }

        case 8:

        {

            return 0;

        }

        }

    }

    return 0;

}

**OUTPUT**

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit2

Enter the data12

12

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit2

Enter the data13

12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit3

Enter the data11

11 12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit6

Enter the data and position10 2

11 10 12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit4

10 12 13

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit5

10 12

1.Display

2.Insert at End

3.Insert at Beginning

4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit7

Enter the position2

10

1.Display

2.Insert at End

3.Insert at Beginning

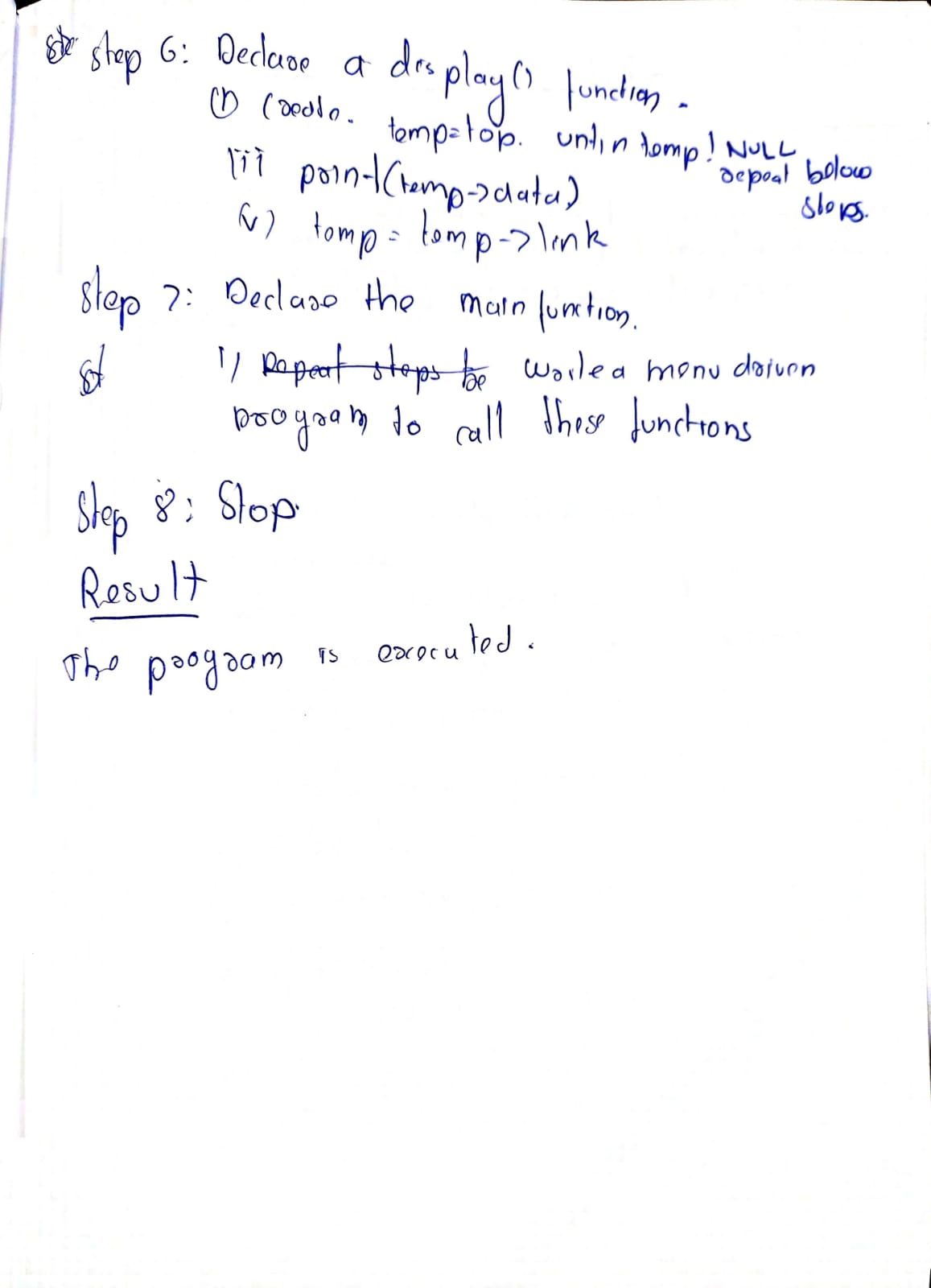
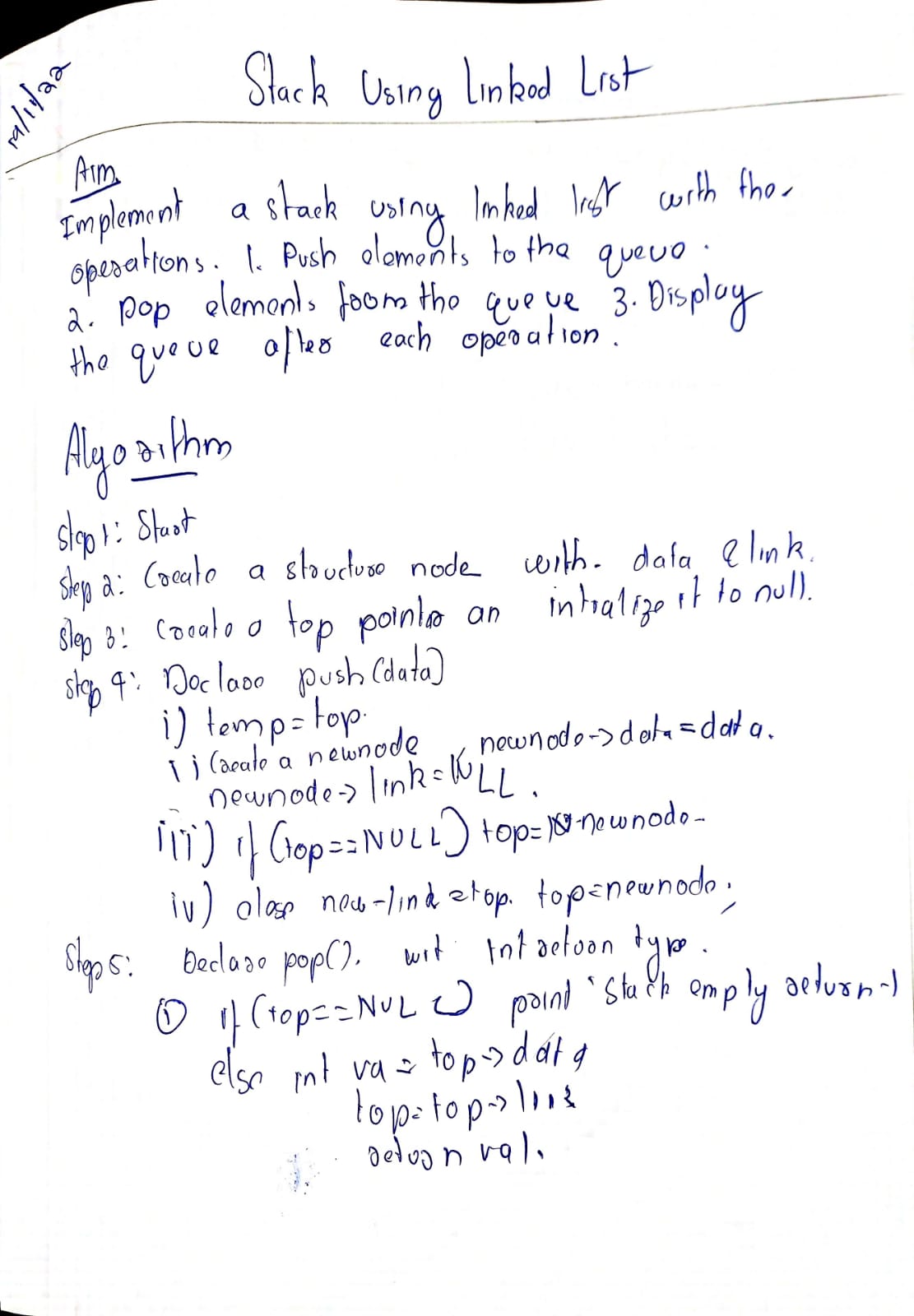
4.Delete from beginning

5.Delete from end

6.Insert into position

7.Delete from position

8.Exit8



**Code**

#include<stdio.h>

#include<stdlib.h>

*struct* node

{

*int* data;

*struct* node\* link;

};

*struct* node\* top=NULL;

*void* push(*int* *data*)

{

*struct* node\* temp=top;

*struct* node\* newnode=(*struct* node\*)malloc(sizeof(*struct* node));

    newnode->data=*data*;

    newnode->link=NULL;

    if(top==NULL)

    {

        top=newnode;

    }else

    {

        newnode->link=top;

        top=newnode;

    }

}

*int* pop()

{

    if(top==NULL)

    {

        printf("Stack is empty");

        return -1;

    }else

    {

*int* val=top->data;

        top=top->link;

        return val;

    }

}

*void* display()

{

    printf("STACK : ");

*struct* node\* temp=top;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->link;

    }

    printf("\n");

}

*int* main()

{

    while(1)

    {

*int* choice;

        printf("1.Push\n2.Pop\n3.Exit\n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

            {

*int* temp;

                printf("Enter the value to push");

                scanf("%d",&temp);

                // add\_end(temp);

                push(temp);

                display();

                break;

            }

            case 2:

            {

*int* temp=pop();

                if(temp!=-1)

                {

                    printf("%d Popped\n",temp);

                }

                display();

                break;

            }

            case 3:

            {

                return 0;

            }

        }

    }

    return 0;

}

**OUTPUT**

1.Push

2.Pop

3.Exit

1

Enter the value to push12

STACK : 12

1.Push

2.Pop

3.Exit

1

Enter the value to push13

STACK : 13 12

1.Push

2.Pop

3.Exit

1

Enter the value to push14

STACK : 14 13 12

1.Push

2.Pop

3.Exit

1

Enter the value to push15

STACK : 15 14 13 12

1.Push

2.Pop

3.Exit

2

15 Popped

STACK : 14 13 12

1.Push

2.Pop

3.Exit

2

14 Popped

STACK : 13 12

1.Push

2.Pop

3.Exit

2

13 Popped

STACK : 12

1.Push

2.Pop

3.Exit

2

12 Popped

STACK :

1.Push

2.Pop

3.Exit

2

Stack is emptySTACK :

1.Push

2.Pop

3.Exit

2

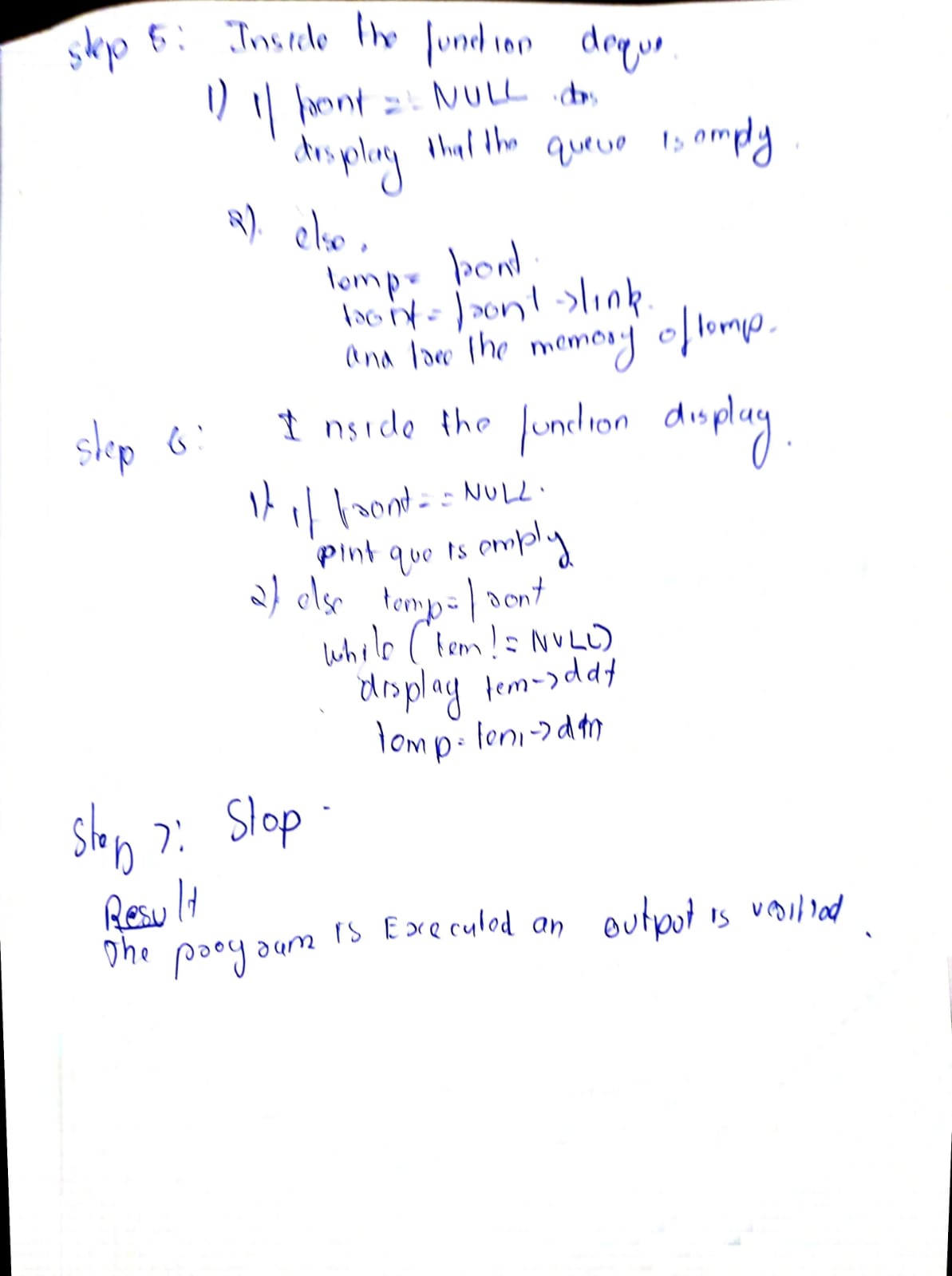
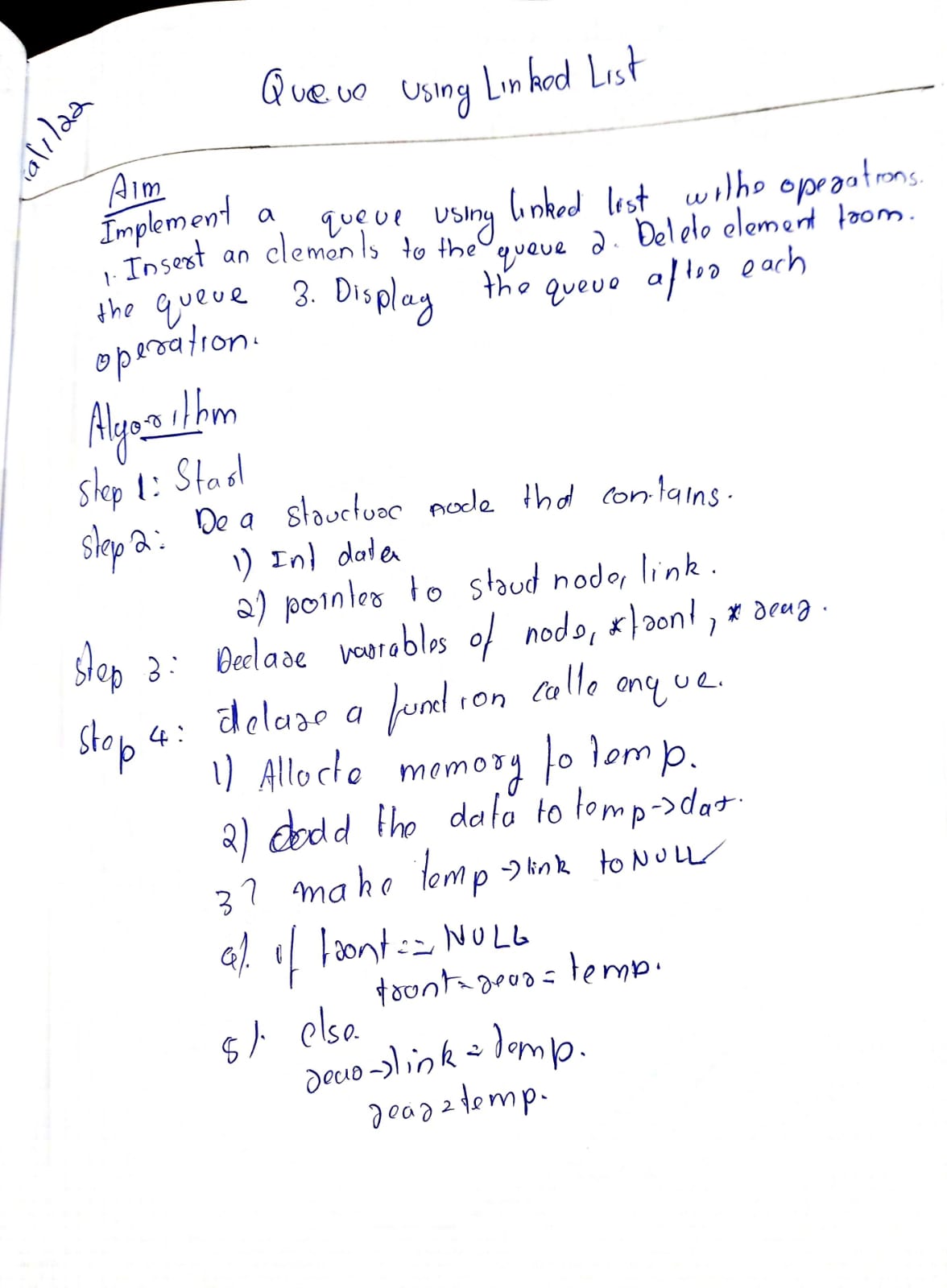
Stack is emptySTACK :

1.Push

2.Pop

3.Exit

3



**Code**

#include<stdio.h>

#include<stdlib.h>

*struct* node

{

*int* data;

*struct* node\* link;

};

*struct* node\* head=NULL;

*struct* node\* front=NULL;

*struct* node\* rear=NULL;

// void display()

// {

//  struct node\* temp=head;

//  while(temp!=NULL)

//  {

//      printf("%d ",temp->data);

//      temp=temp->link;

//  }

//  printf("\n");

// }

*void* enqueue(*int* *data*)

{

*struct* node\* newnode=(*struct* node\*)malloc(sizeof(*struct* node));

    newnode->data=*data*;

    newnode->link=NULL;

    if(front==NULL || rear==NULL)

    {

        front=rear=newnode;

    }else

    {

        rear->link=newnode;

        rear=newnode;

    }

}

*int* dequeue()

{

    if(front==NULL || rear==NULL)

    {

        printf("Queue is empty");

        return -1;

    }else

    {

*int* val=front->data;

        front=front->link;

        return val;

    }

}

*void* display()

{

    printf("QUEUE:\n");

*struct* node\* temp=front;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->link;

    }

    printf("\n");

}

// void enqueue(int data)

// {

//  struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

//  newnode->data=data;

//  newnode->link=head;

//  head=newnode;

// }

// int dequeue()

// {

//  struct node\* temp=head;

//  int val=-1;

//  if(temp==NULL)

//  {

//      printf("The Queue is empty\n");

//  }else

//  {

//      if(head->link!=NULL)

//      {

//          while(temp->link->link!=NULL)

//          {

//              temp=temp->link;

//          }

//          val=temp->link->data;

//          temp->link=NULL;

//      }else

//      {

//          val=head->data;

//          head=NULL;

//      }

//  }

//  return val;

// }

*int* main()

{

    while(1)

    {

*int* choice;

        printf("1.Enqueue\n2.Dequeue\n3.Exit\n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

            {

*int* temp;

                printf("Enter the number to enqueue");

                scanf("%d",&temp);

                enqueue(temp);

                display();

                break;

            }

            case 2:

            {

*int* temp=dequeue();

                if(temp!=-1)

                {

                    printf("%d Dequeued\n",temp);

                }

                display();

                break;

            }

            case 3:

            {

                return 0;

            }

        }

    }

    return 0;

}

**OUTPUT**

1.Enqueue

2.Dequeue

3.Exit

1

Enter the number to enqueue12

QUEUE:

12

1.Enqueue

2.Dequeue

3.Exit

1

Enter the number to enqueue13

QUEUE:

12 13

1.Enqueue

2.Dequeue

3.Exit

1

Enter the number to enqueue14

QUEUE:

12 13 14

1.Enqueue

2.Dequeue

3.Exit

1

Enter the number to enqueue15

QUEUE:

12 13 14 15

1.Enqueue

2.Dequeue

3.Exit

2

12 Dequeued

QUEUE:

13 14 15

1.Enqueue

2.Dequeue

3.Exit

2

13 Dequeued

QUEUE:

14 15

1.Enqueue

2.Dequeue

3.Exit

2

14 Dequeued

QUEUE:

15

1.Enqueue

2.Dequeue

3.Exit

2

15 Dequeued

QUEUE:

1.Enqueue

2.Dequeue

3.Exit

2

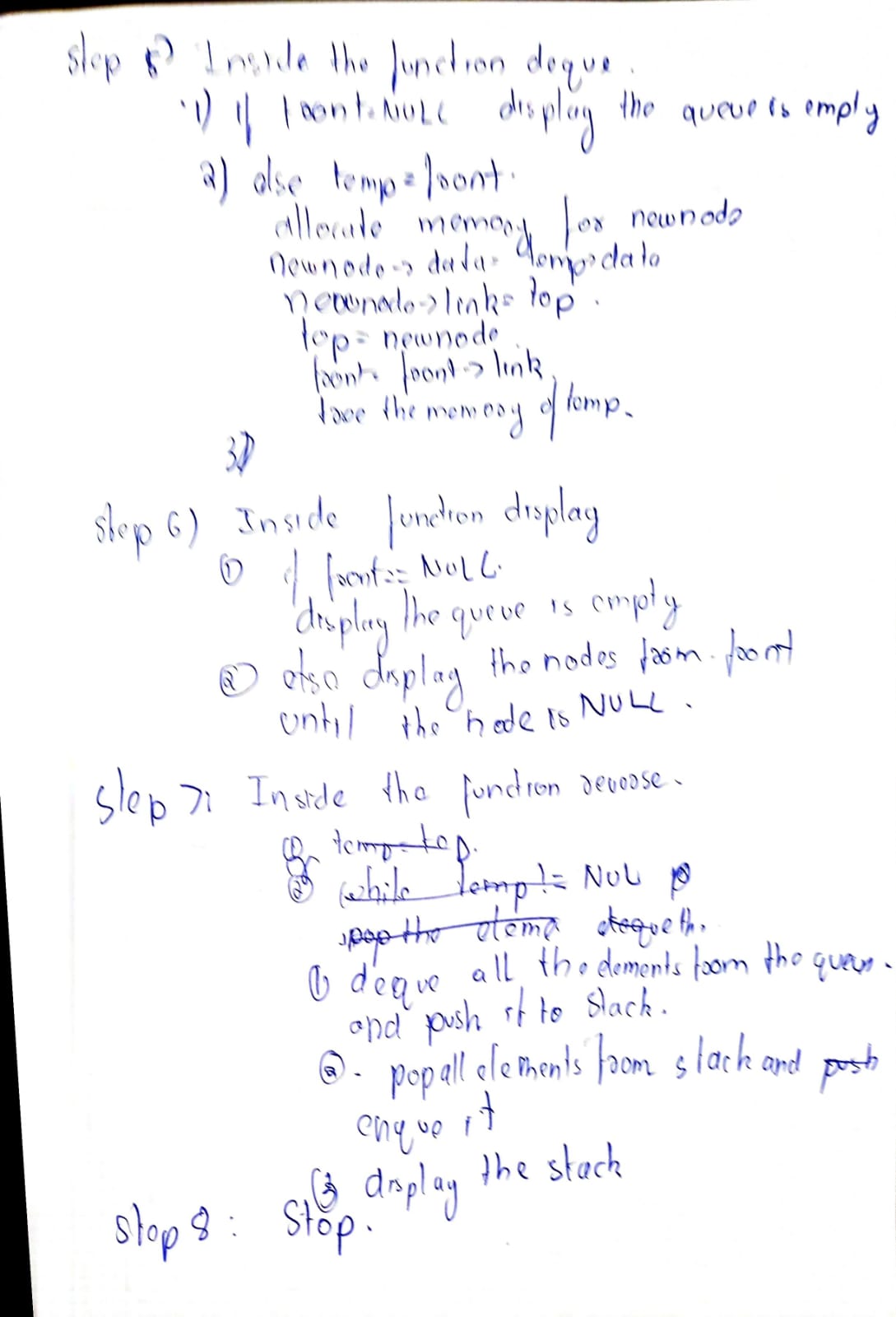
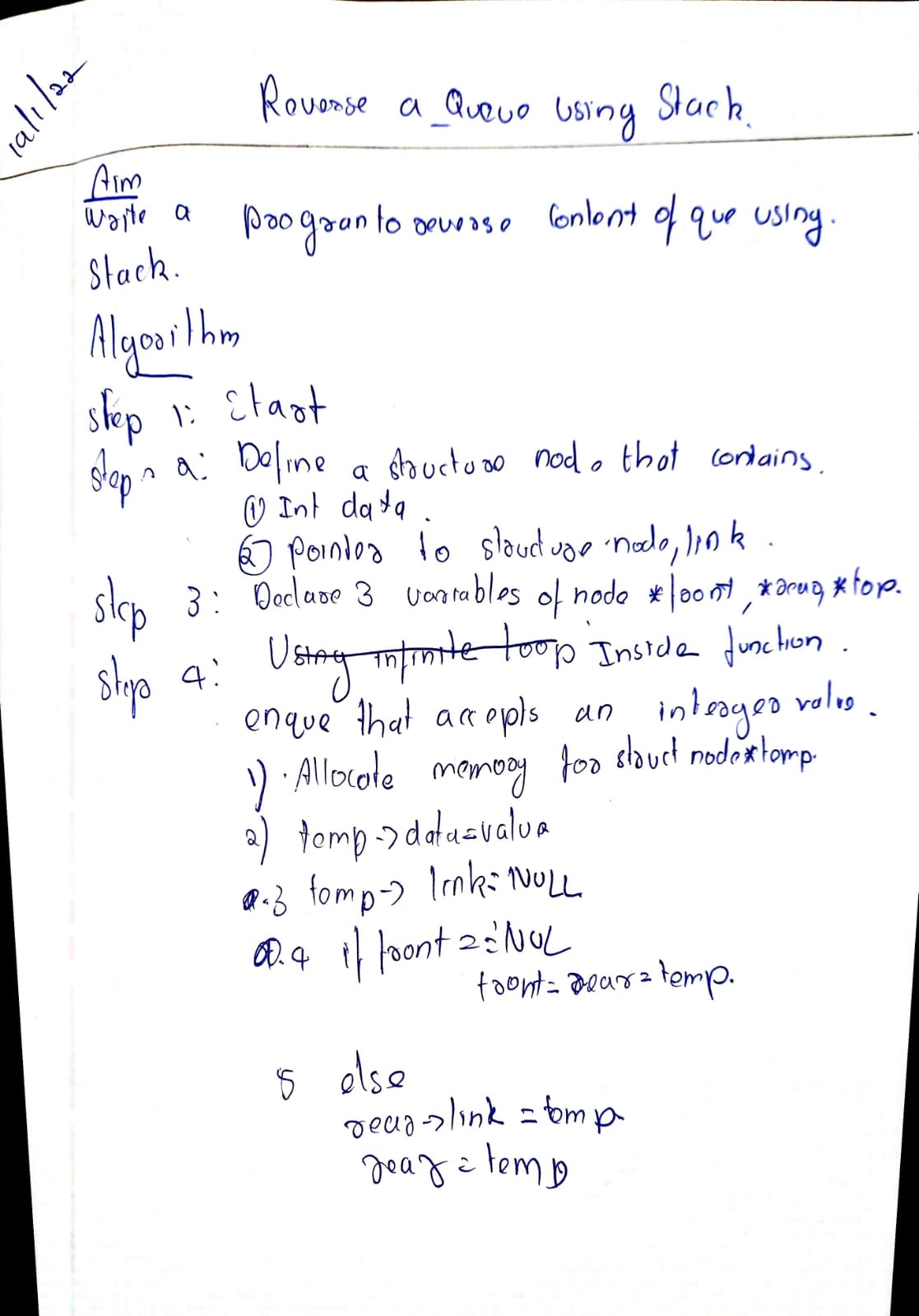
Queue is emptyQUEUE:

1.Enqueue

2.Dequeue

3.Exit

3



**Code**

#include<stdio.h>

#include<stdlib.h>

*struct* node

{

*int* data;

*struct* node\* link;

};

*struct* node\* top=NULL;

*struct* node\* front=NULL;

*struct* node\* rear=NULL;

*void* display()

{

*struct* node\* temp=front;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->link;

    }

    printf("\n");

}

*void* enqueue(*int* *data*)

{

*struct* node\* newnode=(*struct* node\*)malloc(sizeof(*struct* node));

    newnode->data=*data*;

    newnode->link=NULL;

    if(front==NULL || rear==NULL)

    {

        front=rear=newnode;

    }else

    {

        rear->link=newnode;

        rear=newnode;

    }

}

*int* dequeue()

{

    if(front==NULL || rear==NULL)

    {

        printf("Queue is empty");

        return -1;

    }else

    {

*int* val=front->data;

        front=front->link;

        return val;

    }

}

*void* push(*int* *data*)

{

*struct* node\* temp=top;

*struct* node\* newnode=(*struct* node\*)malloc(sizeof(*struct* node));

    newnode->data=*data*;

    newnode->link=NULL;

    if(top==NULL)

    {

        top=newnode;

    }else

    {

        newnode->link=top;

        top=newnode;

    }

}

*int* pop()

{

    if(top==NULL)

    {

        printf("Stack is empty");

        return -1;

    }else

    {

*int* val=top->data;

        top=top->link;

        return val;

    }

}

*void* reverse()

{

    printf("Reversed Queue  ");

    while(front!=NULL)

    {

        push(dequeue());

    }

    while(top!=NULL)

    {

        enqueue(pop());

    }

    display();

    printf("\n");

}

*int* main()

{

    while(1)

    {

*int* choice;

        printf("1.Enqueue\n2.Dequeue\n3.Reverse the Queue\n4.Exit\n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

            {

*int* temp;

                printf("Enter the number to enqueue");

                scanf("%d",&temp);

                enqueue(temp);

                display();

                break;

            }

            case 2:

            {

*int* temp=dequeue();

                if(temp!=-1)

                {

                    printf("%d Dequeued\n",temp);

                }

                display();

                break;

            }

            case 3:

            {

                reverse();

                break;

            }

            case 4:

            {

                return 0;

            }

        }

    }

    return 0;

}

**OUTPUT**

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

1

Enter the number to enqueue12

12

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

1

Enter the number to enqueue13

12 13

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

1

Enter the number to enqueue14

12 13 14

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

1

Enter the number to enqueue15

12 13 14 15

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

1

Enter the number to enqueue16

12 13 14 15 16

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

2

12 Dequeued

13 14 15 16

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

3

Reversed Queue 16 15 14 13

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

4