Q) stack implementation using single linked list;

```
#include <stdio.h>
#include<stdlib.h>
struct node{
        int data;
        struct node * next;
};
struct node *top=0;
void push (int n) {
    struct node *newnode;
newnode =(struct node*) malloc (sizeof (struct node));
    newnode -> data = n;
    newnode -> next = top;
    top = newnode;
}
void display () {
    struct node * temp;
    temp = top;
if (top == 0) {
        printf ("List is Empty);
}
else {
    while (top != 0) {
        printf ("%d", temp -> data);
        temp = temp -> next;
    }
}
void pop () {
    struct node *temp;
    temp=top;
```

```
if (top == 0) {
        printf ("List is empty");
}
else
{
    top = top -> next
    free (temp);
}
}
void main() {
int ch; while (ch != 4) {
printf ("Enter 1: push 2: pop 3: display
                4: exit program);
scanf ("%d", &ch);
switch (ch) {
case 1: int u;
        printf ("Enter value");
        scanf ("%d", &u);
        push(u)
        break;
case 2 : pop()
        break;
case 3 : display()
        break;
case 4 : printf ("Exit");
        break.
default : printf ("Invalid Input")
    }
}
}
```

2) Queue implementation using single linked list.

```
# include <stdio.h>
# include <stdlib.h>
    struct node{
        int data;
        struct node * next;
};
    struct node * front = 0;
    struct node * rear = 0;
        void enqueue (int x) {
        struct node * newnode;
newnode =(struct node *) malloc (size of
            (struct node));
        newnode -> data = x;
        newnode -> next = 0;
if (front == 0 && rear == 0) {
            front = rear = newnode;
    }
else {
    rear -> next = newnode;
        rear = newnode;
    }
}
void display(){
    struct node * temp;
    if (front == 0 && rear == 0){
    printf ("Queue is empty");
}
else {
    temp = front;
    while( temp != 0) {
```

```
        printf("%d", temp->data);
        temp = temp->next;
    }
  }
}

void dequeue(){
    struct node *temp;
        temp = front;
    if(front==0 && rear==0){
        printf("Empty queue");
    }
    else{
        front = front->next
            free(temp);
    }
}
void main(){
    int
    while(ch!=0){
    printf("1. Enqueue, 2. Dequeue, 3. Display
            4. Exit");
    scanf("%d", &ch);
    switch(u){
    case 1: int u
            printf("Enter value")
            Scanf("%d", &u)
            enqueue(u); break;
    case 2: dequeue(); break;
    case 3: display(); break;
    case 0: printf("Exit");
    default: printf("Invalid"); break;
    }
}
}
```

SURYA Gold

Date Page

1.) Sorting, reversing & concatenation:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
        int data;
    struct node * next;
};
void sortList (struct node ** head);
void insertAtBeginning (struct node ** headRef,
  int data) {
  struct node ** newnode;
  newnode = (struct node *) malloc (sizeof (struct node));
  newnode -> data = data;
  newnode -> next = *headRef;
  *headRef = newnode;
}
void printList (struct node *head) {
    while (head != NULL) {
      printf ("%d", head -> data);
      head = head -> next;
}
void sortList (struct node ** head) {
    struct node * current, * nextnode;
    int temp;
    current = *head;
    while (current != NULL) {
    nextnode = current -> next;
    while (nextnode != NULL) {
    if (current -> data > nextnode -> data) {
        temp = current -> data;
```

```
        current -> data = next node -> data
        next node -> data = temp;
    }
    next node = next node -> next;
  }
  current = current -> next;
 }
}

void reverse() {
    struct node * prev node * current node *
        next node;
prevnode = 0;
current node = next node = head;
while ( next node != NULL ) {
    next node = next node -> next;
    current node -> next = prevnode;
    prev node = current node;
    current node = new node;
    head = prev node;
  }
}

void concatenate ( struct node * list1,
                   struct node * list2 ) {
if ( *list1 = NULL ) {
    * list 1 = list 2;
    return;
  }

struct node * temp = * list1;
```

```
while ( temp -> next != NULL) {
        temp = temp -> next ;
}
        temp -> next = list2 ;
}

int main () {
    struct node * list 1 = NULL ;
    struct node * list 2 = NULL ;
    int choice ;
    int data ;
    while (1) {
      Pf (1. Insert list 1 );
      pf  (2. Insert lis 2 );
      Pf (3. Dcart lis 1 ) ;
      pf ( 4. Reverse list 2 );
      pf (5. Concatenate List ");
      pf ( 6. Print List ");
      Pf ( 0. Exit . program .
      Scanf ("%d" & choice );

Case 1)  "Enter data"
          insert At Beginning (& list 1 , & list 2)

case 2 :  "Enter data"
          insert At Beginning ( & list 2, data)

case 3 :  Dent list (& list n )
case 4 :  Reverse ( )
case 5 :  Concate list ( & list , & list 2 )
```

Output :

Enter data to insert into list : 1 2 3 4 5

list reversed

reversed list : 5 4 3 2 1.

data : 1 3 6 8 11 9 11 2.

list sorted : 1 2 3 4 5 6 8 9 11

2) Enter : (1). push (2) pop (3) display
(4) exit prog.

Enter value 5.

push(5)

push(2)

pop()

display()

5 3.

3) Enter (1) enqueue (2) dequeue (3) display
(4) Exit prog.

Enter value 6

enqueue(4)

enqueue(1)

Sir
5/12/24.

Enter (1) enque (2) dequeue (3) display
(0) exit):

6 4 1

dequeue()

dequeue()

display()

1.