Week-8 (29/01/24)

sll_sort, reverse, concatination

1) Concatinate ⇒

```
struct node *concatinate (struct node *list 1, struct
                                  node *list 2)
{ if (list 1 == list 2){
      return list 2;
  }
  struct node *ptr = list 1;
  while (ptr → next != NULL){
            ptr = ptr → next;
  }
    ptr → next = list 2;
      return list 2;
}
```

2) Sort ⇒

```
void sortlist (struct node **head){
      struct node *ptr, *nodenext;
      int temp;
      ptr = *head;
      while (ptr != NULL){
        node next = ptr → next;
          while (nodenext != NULL){
                if (ptr → data > nodenext → data){
                  temp = ptr → data;
                  ptr → data = nodenext → data;
                  nodenext → data = temp;
                }
                nodenext = nodenext → next;
          }
      } ptr = ptr → next }
```

(2) <u>Reverse</u> ⇒

```
void reverse (struct node **head) {
    struct node *prev, *current, *nextnode;

    prev = NULL;
    current = *head;
    while (current != NULL){
        nextnode = current→next;
        current→next = prev;
        prev = ~~curr~~ current;
        current = nextnode;
    }
    *head = prev;
}
```

(II) (a) Implement stack using LL.

```
⇒ ~~void~~ struct stack{
        struct node *top;
    };
    void init_stack (struct stack *stack) {
        { stack → top = NULL;
    }
 void push (struct stack, *stack, int value)
    { struct node * newnode = create node (value);
        newnode → next = stack → top;
        stack → top = newnode;
    }
int pop (struct stack* stack){
    if (isEmpty (stack)){
            pf(" Underflow");
        exit(1);
    }
    int pop_v = stack→top → data;
    struct node* temp = stack → top;
    stack → top = stack → top → next;
    free (temp);
    return pop_v;
  }
```

(b) Implement queue using single linked list

```
void Insert (struct queue * queue, int value){
    struct node * newnode = createnode (value);
    if (isempty (queue)){
            queue → front = newnode;
            queue → rear = newnode;
    }
    else {
        queue→rear→next = newnode;
        queue → rear = newnode;
    }
}
int dequeue (struct * queue *queue){
    if (isempty(queue)){
            pf(" underflow");
    }
    int dequeued_value = queue → front → data;
    struct node * temp = queue → front;
    if (queue → front = queue → rear){
        queue → front = NULL;
        queue → rear = NULL; }
    else {
        queue → front = queue→front → next;
    }
free (temp);
```

o/p ⇒

① Enter your choice :

```
1. create alist1  2. create list2  3. concatenate  4. print
5. exit
Enter choice : 1
Enter data : 3 → 5 → 1 → NULL
Enter choice : 2
Enter data : 6 → 7 → 8 → NULL
Enter choice : 3
  List 1 sorted
  enter choice : 4
  List 1 reversed
  1 → 5 → 3 → NULL
Enter choice : 5
  concatenated
Enter choice : 6
  3 → 5 → 1 → 6 → 7 → 8 → NULL
```

② o/p ⇒

```
Stack after pushing elements.
30 → 20 → 10 → NULL
Poped value : 30
stack after popping element :
20 → 10 → NULL
Top value : 20
Top value before popping = 30
```

③ o/p ⇒

```
Queue after enqueing elements
10 ← 20 ← 30 ← NULL
Dequeue value ⇒ 10
Queue after dequeuing element :
20 ← 30 ← NULL.
```