21/01/2024 | WEEK - 8 |

SLL - sort, reverse, concatenation

P-1) 1) Concatenate ⇒

```
struct node* conca_lts (struct node *list1, struct node *list2)
{      if (list1 ==NULL) {
            return list2;
       }
       struct node *ptr = list1;
       while (ptr->next != NULL) {
               ptr = ptr->next;
       }
       ptr->next = list2;
       return list1;
}
```

2) Sort ⇒

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

```
3) reversed   void reverse (struct node **head) {
                  struct node *prev, *current, *nextNode;
                  prev = NULL;
                  current = *head;
                  while (current != NULL) {
                        nextNode = current->next;
                        current->next = prev;
                        prev = current;
                      } current = nextNode;

              } *head = prev;
```

II) a) Implement stack using linked lists.

```
Pr    ~~void initialisestack~~ (~~struct Stack* stack~~)
        struct Stack {
              struct node *top;
          };
      void initstack (struct stack *stack)
        {     stack->top = NULL;
          }
      void push (struct Stack* stack, int value)
        {    struct node *newnode = createnode(value);
               newnode->next = stack->top;
               stack->top = newnode;
          }
        int pop (struct Stack* stack) {
              if (isEmpty(stack)) {
                  printf("Underflow*\n");
                  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 deque(struct Queue* queue)
  {
          if (isEmpty (queue))
          {  printf ("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);
          return(dequeued value)
  }
```

```
o/p:- Enter your choice:
I) 1) 1.create list1  2.create list 2  3.concatenate ~~3~~ 4 ~~exit~~ print
                                                  5 ~~6~~. exit
 1 ~~ent~~ enter -1 to exit:
     enter data: 1
     enter data: 2
     enter data: -1
 Enter your choice:-
 1. create list1  2. create list 2  3. concatenate ~~3~~ 4. print
                                                  5 ~~4~~. exit
 2
     enter -1 to exit:
      enter data: 3
      enter data: 4
      enter data: 5
      enter data: -1
  Enter your choice:-
 1. create list1   2. create list 2  3. concatenate ~~3~~ 4. print
                                                   5 ~~4~~. exit
  3
     Lists are concatenated
  Enter your choice:-
 1. create list1   2. create list 2  3. concatenate ~~3~~ 4. print
                                                   5 ~~4~~. exit
  ~~3~~ 4

     1  2  3  4  5
     1  2
     3  4  5
2) Enter choice:-
   1. create   2. sort   3. print   4. exit
   1
     enter -1 to exit:
      enter data: 5
      enter data: 2
      enter data: 3
      enter data: 9
  Enter choice:-
   1. create   2. sort   3. print   4. exit
   2

  Enter choice:-
  1. create   2. sort   3. print  4. exit

  3
   2 3 5 9
3) Enter choice:-
   1. create   2. ~~sort~~ reverse  3. print  4. exit
   1
    enter -1 to exit:
     enter data: 1
     enter data: 2
     enter data: 0
     enter data: -1
  Enter choice:-
   1. create   2. reverse   3. print   4. exit
   2
  Enter choice:-
   1. create   2. reverse   3. print   4. exit
   3
   0  2  1
II) a) Enter choice:-
   1) push   2) pop   3) display   4) exit
   1 ~~enter elements~~
   No. of elements to be pushed: 3
    element 1:  50
    element 2:  60
    element 3:  70
  Enter choice:-
  1) push   2) pop   3) display  4) exit
  3
    Stack is   70   60   50
  Enter choice:-
  1) push   2) pop   3) display   4) exit
```

3) o/p:

```
Enter choice: 1)enqueue 2)deque 3)display
        4)Exit
1.

Enter -1 to exit:
Enter data: 1
Enter data: 2
Enter data: 3
Enter data: -1
Enter choice: 1)enqueue 2)deque 3)display
                4)Exit

3

1   2   3
Enter choice: 1)enque 2) deque 3) display 4)Exit

2
Enter choice: 1)enque 2)deque 3)display 4)Exit

2 3
```

✓

[illegible]
5/2/24

05/02/2024

WEEK - 07