29-01-2024

Prog-07.

07) Sort operation on Single linked list

```
# include <stdio.h>
# include <stdlib.h>
Struct Node
{
 int data;
 struct Node * next;
};
Void insertNode (Struct Node ** head, int newData)
{
Struct Node * newNode = (Struct Node *) malloc (size of (Struct Node));
New Node -> data = newData;
new Node -> next = NULL;
if (* head == NULL)
{
 * head = newNode;
}
else
{
Struct Node * temp = * head;
while (temp -> next != NULL)
{
 temp = temp -> next;
}
temp -> next = newNode;
}
}
void printList (Struct Node * head)
{
 Struct Node * temp = head;
```

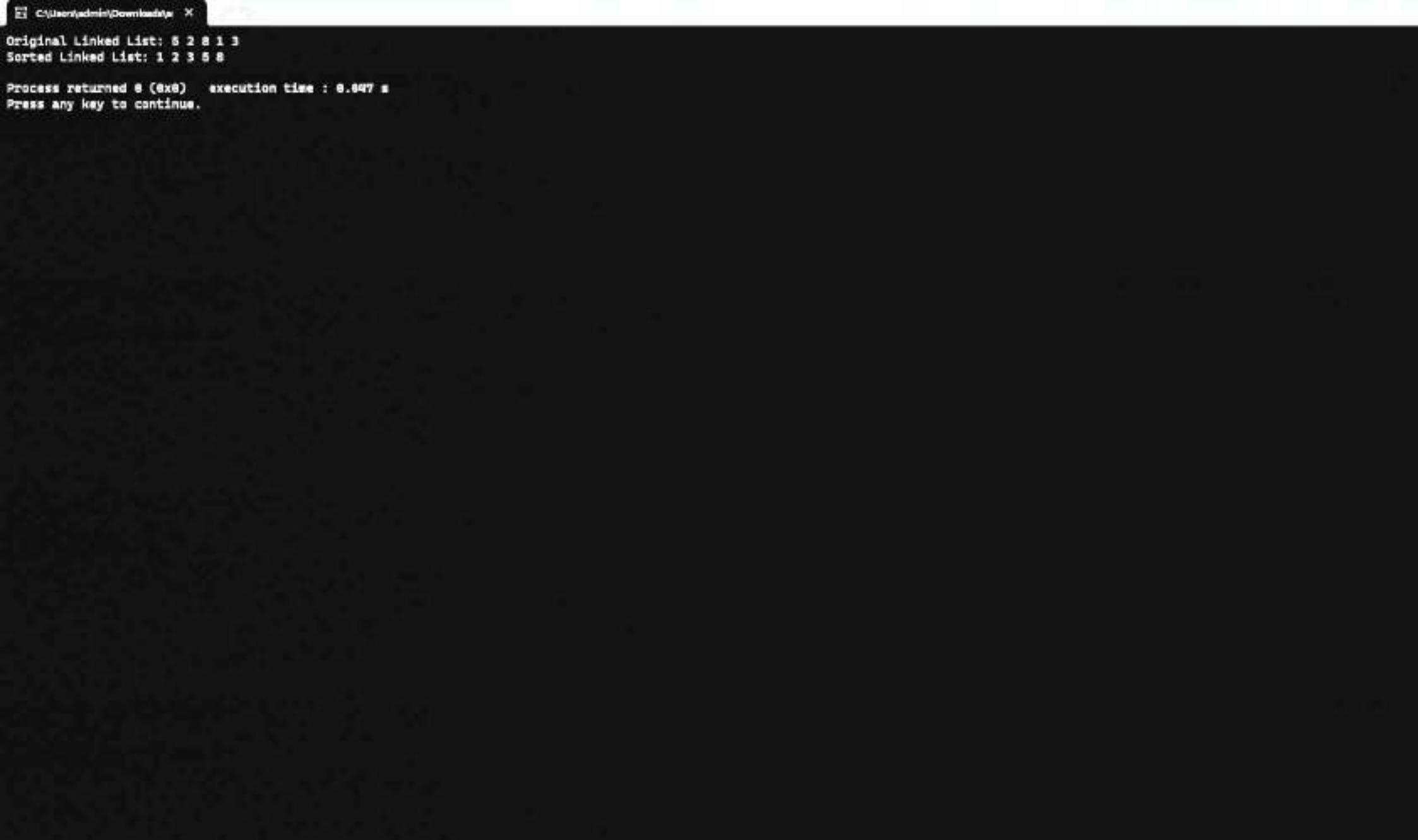
29-01-2024

```
while (temp != NULL)
{
print ("%d", temp -> data);
 temp = temp -> next;
}
printf ("\n");
}
void bubblesort (struct Node * head)
{
int swapped, i;
struct Node * ptr;
struct Node * lptr = NULL;
if (head == NULL)
{
 return;
}
do
{
Swapped = 0;
ptr = head;
while (ptr -> next != lptr)
{
if (ptr -> data > ptr -> next -> data)
{
 int temp = ptr -> data;
 ptr -> data = ptr -> next -> data;
 ptr -> next -> data = temp;
 swapped = 1;
}
ptr = ptr -> next;
}
```

29-01-2024

```
    dptr = ptr;
    }
  while (swapped);
  }
int main()
  {
  struct Node * head = NULL;
  insertNode (& head, 5);
  insert Node (& head, 2);
  insert Node (& head, 8);
  insert Node (& head, 1);
  insert Node (& head, 3);
  printf ("original linked list:");
  printlist (head);
  bubblesort (head);
  printf ("sorted linked list :");
  printlist (head);
  return 0;
  }
```

o/p original linked list : 5 2 8 1 3
Sorted linked list : 1 2 3 5 8


C:\Users\admin\Downloads\a
Original Linked List: 5 2 8 1 3
Sorted Linked List: 1 2 3 5 8
Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.

29-01-2024 prog-087

08) Reverse operation on single linked list

```
#include <stdio.h>
#include <stdio.h>
Struct Node
{
int data;
struct Node* next;
};
Stud Node * CreateNode (int value)
{
Struct Node * newNode = (StructNode *) malloc (size of (Struct Node))
if (newNode == NULL)
{
printf ("Memory allocation failed \n");
exit (1);
}
NewNode -> data = value;
New Node -> next = NULL;
return newNode;
}
Void insert End (struct node ** head, int value)
{
Struct Node* temp = (Struct Node *) malloc (size of (Stud Node));
temp -> data = data;
temp -> next = NULL;
return temp;
}
int main()
{
Struct Node * head = newNode (1);
```

29-01-2024

```
head -> next = newNode(2);
head -> next -> newNode(3);
head -> next -> next -> next = newNode(4);
head -> next -> next -> next -> next = newNode(5);
printf("original linked list:");
printlist(head);
head = reverse linked list(head);
printf("Reversed linked list:");
printlist(head);
return 0;
}
```

o/p

```
original linked list : 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Reversed linked list : 5 -> 4 -> 3 -> 2 -> 1 -> NULL
```

```
Original Linked List: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Reversed Linked List: 5 -> 4 -> 3 -> 2 -> 1 -> NULL

Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```

29-01-2024

Prog - 07

09 Concatenation operation on single linked list

```
#include<stdio.h>
#include<stdio.h>
StructNode
{
int data;
Struct Node* next;
};
Struct Node* create Node (int data)
{
Struct Node *newNode = (Struct Node*) malloc (sizeof (StructNode));
newNode -> data = data;
newNode -> next = NULL;
return new Node;
}
void display list (struct Node* head)
{
Struct Node* current = head;
while (current != NULL)
{
printf ("%d ->", current -> data);
current = current -> next;
}
printf ("NULL\n");
}
Struct Node * concatenate lists (struct Node * list1, struct Node*
                                 list2)
{
if (list1 == NULL)
{
  return list2;
}
```

29-01-2024

```
Struct Node * current = list1;
while (current -> next != NULL)
{
current = current -> next;
}
current -> next = list2;
return list1;
}
int main()
{
Struct Node * list1 = CreateNode(1);
list1 -> next = Create Node(2);
list2 -> next -> next = CreateNode(3);
StructNode * list2 = Create Node(4);
list2 -> next = Create Node(5);
list2 -> next -> next = CreateNode(6);
printf("First linked list:");
displayList(list1);
Printf("Second linked list:");
display list(list2);
Struct Node * Concatenated list = Concatenate lists(list1, list2);
printf("Concatenated linked list:");
display list(Concatenated list);
return 0;
}
```

o/p

First linked list : 1 -> 2 -> 3 -> NULL
second linked list : 4 -> 5 -> 6 -> NULL
Concatenated linked list : 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> NULL

```
First Linked List: 1 -> 2 -> 3 -> NULL
Second Linked List: 4 -> 5 -> 6 -> NULL
Concatenated Linked List: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> NULL

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

29-01-2024

Prog-08

Q Stack implementation using single linked list

=>
```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
int data;
struct Node * next;
};
Struct Node * Create Node (int data)
{
Struct Node * newNode = (struct Node *) malloc (sizeof (struct Node));
if (newNode == NULL)
{
printf ("Memory allocation failed \n");
exit (exit_Failure);
}
newNode -> data = data;
newNode -> next = NULL;
return newNode;
}
Struct Node * pop (struct Node *tp)
{
if (top == NULL)
{
Printf ("Stack underflow = Cannot pop \n");
Return NULL;
}
Struct Node * temp = top;
top = top -> next;
free (temp);
return top;
}
```

29-01-2024

```
void displaystack(struct Node * top);
{
printf("stack:");
while(top!=NULL)
{
printf("%d", top->data);
top=top->next;
}
printf("\n");
}
void freestack(struct Node * top)
{
while(top!=NULL)
{
struct Node * temp = top;
top=top->next;
free(temp);
}
}
int main()
{
struct Node * top=NULL;
int choice, data;
do
{
printf("\n Menu:\n");
printf("1. push \n");
printf("2. pop \n");
printf("3. Display \n");
printf("4. Exit \n");
```

29-01-2024

```
printf (" Enter your choice :");
scanf ("%d", &choice);
switch (choice)
{
case 1 :
   printf ("Enter data to push");
   scanf ("%d", &data);
   top= push (top, data);
   break;

case 2 :
   top= pop (top);
   break;

case 3 :
   display Stack (top);
   breaks;

case 4 :
   printf (" Entering the program \n");
   break;

default:
   Printf (" Invalid choice ");
 }
}
while (choice != 4)
freestack (top);
return 0;
}
```

29-01-2024

o/p Menu

1. push
2. pop
3. Display
4. Exit

Enter your choice : 1
Enter data to push : 6
Enter your choice : 2
popped successfully
Enter your choice : 1
Enter data to push : 7
Enter your choice : 3
7. 6.

```
10 pushed to the stack.
20 pushed to the stack.
30 pushed to the stack.
Stack elements: 30 20 10
Top element: 30
Popped element: 30
Stack elements: 20 10

Process returned 0 (0x0)   execution time : 0.082 s
Press any key to continue.
```

29-01-2024 prog-09.

Q Queue implementation using linked list.

```
#include<stdio.h>
#include<stdlib.h>
Struct Node
{
int data
Struct Node * next;
};
Struct Queue
{
Struct node * front;
Struct Node * rear;
};
Struct Node * Create node (int data)
{
Stud Node * newNode = (Struct Node *) malloc (sizeof ( Struct Node));
if (newNode = NULL)
{
printf ("Memory allocation failed");
Exit (Exit - Failure);
}
newNode -> data = data;
new Node -> next = NULL;
return newNode;
}
Struct Queue * initialize Queue()
{
Struct Queue * queue = (Struct Queue*) malloc (size of (Struct queue))
if (queue = NULL)
{
```

29-01-2024

```
printf("Memory Allocation failed");
exit(Exit_failure);
}
queue->front = queue->rear = NULL;
return queue;
}
void enqueue(struct Queue* queue, int data)
{
 Struct Node* newNode = CreateNode(data);
 if(queue->rear = NULL)
{
queue->front = queue->rear = newNode;
return;
queue->rear->next = newNode;
queue->rear = newNode;
}
void dequeue(Struct queue* queue)
{
if(queue->front == NULL)
{
printf("underflow - Cannot dequeue\n");
return;
}
Struct Node* temp = queue->front;
queue->front = queue->front->next;
if(queue->front == NULL)
{
queue->rear = NULL;
}
free(temp);
}
```

29-01-2024

```
void display_queue (struct queue* queue)
{
if (queue->front == NULL)
{
printf ("Queue is empty \n");
return;
}
struct Node * current = queue -> front;
printf ("Queue");
while (current != NULL)
{
printf ("%d", current -> data);
current = current -> next;
}
printf ("\n");
}
int main()
{
struct Queue * queue = initialize_queue();
int choice data;
do
{
printf ("Menu");
printf ("1.Enqueue \n");
printf (" 2. Dequeue \n");
printf ("3. Display \n");
printf ("4. Exit \n");
printf (" Enter Your choice");
scanf ("%d", &choice);
switch (choice)
{
```

29-01-2024

```
case 1:
printf ("Enter data to enqueue:");
scanf ("%d", &data);
Enqueue (queue, data);
break;

case 2:
dequeue (Queue);
break;

case 3:
display Queue (queue);
break;
case 4:
printf ("Exiting the program \n");
break;
default:
printf ("Invalid choice!");
}
while (choice != 4)
}
}
return 0;
}
```

O/p

```
Menu.
1. Enqueue
2. Dequeue
3. Display
4. Exit.
```

29-01-2024

Enter your choice : 3

Queue is Empty

Enter your choice : 1

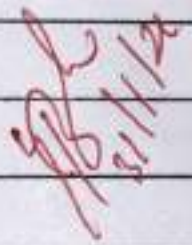
Enter data to enqueue : 4

Enter your choice : 1

Enter data to enqueue : 6

Enter your choice : 3

4 6



```
Queue: 10 20 30
Dequeued element: 10
Queue: 20 30

Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```