**Priority Queue**

**Code:**

// C code to implement Priority Queue

// using Linked List

#include <stdio.h>

#include <stdlib.h>

// Node

typedef struct node {

int data;

// Lower values indicate higher priority

int priority;

struct node\* next;

} Node;

// Function to Create A New Node

Node\* newNode(int d, int p)

{

Node\* temp = (Node\*)malloc(sizeof(Node));

temp->data = d;

temp->priority = p;

temp->next = NULL;

return temp;

}

// Return the value at head

int peek(Node\*\* head)

{

return (\*head)->data;

}

// Removes the element with the

// highest priority form the list

void pop(Node\*\* head)

{

Node\* temp = \*head;

(\*head) = (\*head)->next;

free(temp);

}

// Function to push according to priority

void push(Node\*\* head, int d, int p)

{

Node\* start = (\*head);

// Create new Node

Node\* temp = newNode(d, p);

// Special Case: The head of list has lesser

// priority than new node. So insert new

// node before head node and change head node.

if ((\*head)->priority > p) {

// Insert New Node before head

temp->next = \*head;

(\*head) = temp;

}

else {

// Traverse the list and find a

// position to insert new node

while (start->next != NULL &&

start->next->priority < p) {

start = start->next;

}

// Either at the ends of the list

// or at required position

temp->next = start->next;

start->next = temp;

}

}

// Function to check is list is empty

int isEmpty(Node\*\* head)

{

return (\*head) == NULL;

}

// Driver code

int main()

{

// Create a Priority Queue

// 7->4->5->6

Node\* pq = newNode(4, 1);

push(&pq, 5, 2);

push(&pq, 6, 3);

push(&pq, 7, 0);

while (!isEmpty(&pq)) {

printf("%d ", peek(&pq));

pop(&pq);

}

return 0;

}

**Output:**

7 4 5 6