```python
from collections import deque
from datetime import datetime

class Task:
  def __init__(self, name, due_date, priority):
    self.name = name
    self.due_date = due_date
    self.priority = priority
  def __str__(self):
    priority_map = {1: "High", 2: "Medium", 3: "Low"}
    pr_text = priority_map.get(self.priority, str(self.priority))
    return f"Task: {self.name} | Due Date: {self.due_date.date()} | Priority: {pr_text}"

class Node:
  def __init__(self, task):
    self.task = task
    self.next = None

class LinkedList:
  def __init__(self):
    self.head = None
  def append(self, task):
    new_node = Node(task)
    if not self.head:
      self.head = new_node
    else:
      current = self.head
      while current.next:
        current = current.next
      current.next = new_node

class TaskManager:
  def __init__(self):
    self.tasks = []
    self.completed = LinkedList()
    self.urgent = deque()

  def add_task(self):
    name = input("Enter task name: ")
    date_str = input("Enter due date (YYYY-MM-DD): ")
    try:
      due_date = datetime.strptime(date_str, "%Y-%m-%d")
    except ValueError:
      print("Invalid date format.")
      return
    try:
      priority = int(input("Enter priority (1=High, 2=Medium, 3=Low): "))
    except ValueError:
      print("Priority must be a number.")
      return
    task = Task(name, due_date, priority)
    self.tasks.append(task)
    print("Task added successfully.")

  def view_tasks(self):
    if not self.tasks:
      print("No tasks available.")
      return
    print("Main Task List:")
    for i, task in enumerate(self.tasks, 1):
      print(f"{i}. {task}")
```

```
def delete_task(self):
    self.view_tasks()
    if not self.tasks:
        return
    try:
        idx = int(input("Enter the number of the task to delete: "))
    except ValueError:
        print("Invalid input.")
        return
    if 1 <= idx <= len(self.tasks):
        removed = self.tasks.pop(idx-1)
        print(f"Deleted task: {removed.name}")
    else:
        print("Invalid task number.")

def sort_by_priority(self):
    self.tasks.sort(key=lambda t: t.priority)
    print("Tasks sorted by priority.")

def sort_by_date(self):
    self.tasks.sort(key=lambda t: t.due_date)
    print("Tasks sorted by due date.")

def complete_task(self):
    self.view_tasks()
    if not self.tasks:
        return
    try:
        idx = int(input("Enter the number of the task to mark as completed: "))
    except ValueError:
        print("Invalid input.")
        return
    if 1 <= idx <= len(self.tasks):
        completed_task = self.tasks.pop(idx-1)
        self.completed.append(completed_task)
        print("Task marked as completed and moved to completed list.")
    else:
        print("Invalid task number.")

def view_completed(self):
    if not self.completed.head:
        print("No completed tasks.")
        return
    print("Completed Tasks:")
    current = self.completed.head
    num = 1
    while current:
        print(f"{num}. {current.task}")
        current = current.next
        num += 1

def add_urgent(self):
    name = input("Enter urgent task name: ")
    date_str = input("Enter due date (YYYY-MM-DD): ")
    try:
        due_date = datetime.strptime(date_str, "%Y-%m-%d")
    except ValueError:
        print("Invalid date format.")
        return
    try:
            priority = int(input("Enter priority (1=High, 2=Medium, 3=Low): "))
        except ValueError:
            print("Priority must be a number.")
            return
        task = Task(name, due_date, priority)
        self.urgent.append(task)
        print("Urgent task added to queue.")

    def view_urgent(self):
if not self.urgent:
            print("No urgent tasks.")
            return
        print("Urgent Tasks:")
        for i, task in enumerate(self.urgent, 1):
            print(f"{i}. {task}")

def main():
    manager = TaskManager()

    manager.tasks.append(Task("Finish project report", datetime(2025, 8, 30), 1))
    manager.tasks.append(Task("Study for exam", datetime(2025, 8, 28), 2))
    manager.tasks.append(Task("Buy groceries", datetime(2025, 8, 27), 3))
    manager.tasks.append(Task("Call client", datetime(2025, 8, 26), 1))
    manager.tasks.append(Task("Clean house", datetime(2025, 8, 29), 3))

    manager.urgent.append(Task("Submit assignment", datetime(2025, 8, 25), 1))

    while True:
        print("\nSelect an option:")
        print("1. Add new task")
        print("2. View all tasks")
        print("3. Delete a task")
        print("4. Sort tasks by priority")
        print("5. Sort tasks by due date")
        print("6. Complete a task")
        print("7. View completed tasks")
        print("8. Add urgent task")
        print("9. View urgent tasks")
        print("0. Exit")
        choice = input("Enter your choice: ")
        if choice == "1":
            manager.add_task()
        elif choice == "2":
            manager.view_tasks()
        elif choice == "3":
            manager.delete_task()
        elif choice == "4":
            manager.sort_by_priority()
        elif choice == "5":
            manager.sort_by_date()
        elif choice == "6":
            manager.complete_task()
        elif choice == "7":
            manager.view_completed()
        elif choice == "8":
            manager.add_urgent()
        elif choice == "9":
            manager.view_urgent()
        elif choice == "0":
            print("Exiting program.")
            break
        else:
            print("Invalid choice, try again.")

if name == "__main__":
    main()
```