Name: Saima Ashfaque Amber

Roll no:C-65

Subject: DS

Problem Statement: Simulate a print queue system using linked lists.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

/*STRUCTURE DEFINITIONS*/

/* Node structure to represent each print job */
struct Node {
   int jobId;
   char documentName[100];
   int pageCount;
   struct Node* next;
};

/* Queue structure having pointers to front and rear */
struct Queue {
   struct Node* front;         /* Points to first job */
   struct Node* rear;          /* Points to last job */
```

```
};

/* FUNCTION DECLARATIONS*/
struct Queue* createQueue(void);
struct Node* createNode(int id, char* name, int pages);
void enqueue(struct Queue* q, int id, char* name, int pages);
void dequeue(struct Queue* q);
void display(struct Queue* q);
void displayMenu(void);


/* Function to create and initialize an empty queue */
struct Queue* createQueue(void)
{
   struct Queue* q;
   q = (struct Queue*) malloc(sizeof(struct Queue));
   q->front = NULL;
   q->rear = NULL;
   return q;
}

/* Function to create a new node (print job) */
struct Node* createNode(int id, char* name, int pages)
{
   struct Node* newNode;
   newNode = (struct Node*) malloc(sizeof(struct Node));

   /* Assign values */
   newNode->jobId = id;
   strcpy(newNode->documentName, name);
   newNode->pageCount = pages;
   newNode->next = NULL;

   return newNode;
}

/* Function to add a new print job to the queue (Enqueue) */
void enqueue(struct Queue* q, int id, char* name, int pages)
```

```
{
    struct Node* newNode;
    newNode = createNode(id, name, pages);

    /* If queue is empty, new node becomes both front and rear */
    if (q->rear == NULL) {
        q->front = newNode;
        q->rear = newNode;
        printf("\nJob %d (%s) added to the print queue.\n", id, name);
        return;
    }

    /* Add the new node at the end and change rear */
    q->rear->next = newNode;
    q->rear = newNode;

    printf("\nJob %d (%s) added to the print queue.\n", id, name);
}

/* Function to process and remove a job from the queue (Dequeue) */
void dequeue(struct Queue* q)
{
    struct Node* temp;

    /* Check if queue is empty */
    if (q->front == NULL) {
        printf("\nNo jobs in queue to process.\n");
        return;
    }

    /* Store front node temporarily */
    temp = q->front;

    /* Display job details being processed */
    printf("\nPrinting Job ID: %d\n", temp->jobId);
    printf("Document Name : %s\n", temp->documentName);
```

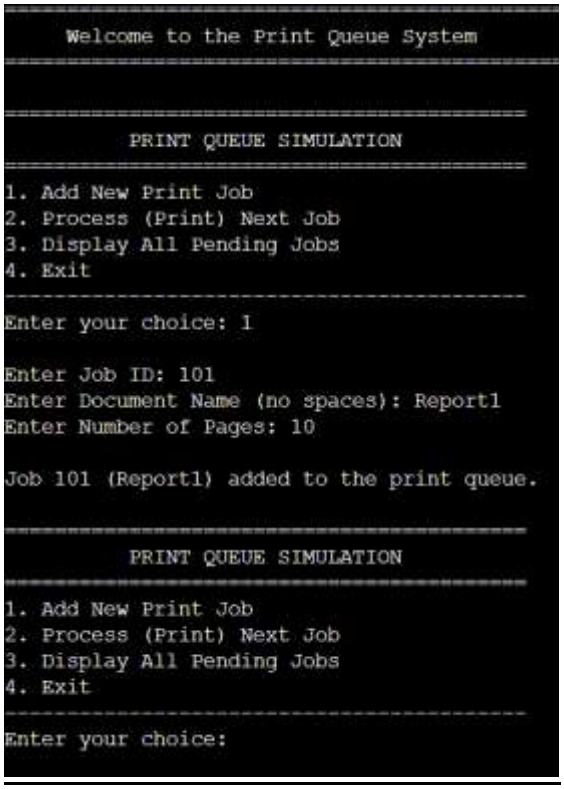
```
    printf("Pages       : %d\n", temp->pageCount);
    printf("Status      : Completed\n");

    /* Move front to next node */
    q->front = q->front->next;

    /* If queue becomes empty after dequeue, set rear = NULL */
    if (q->front == NULL)
        q->rear = NULL;

    /* Free memory of processed job */
    free(temp);
}

/* Function to display all pending jobs */
void display(struct Queue* q)
{
    struct Node* temp;

    /* Check if queue is empty */
    if (q->front == NULL) {
        printf("\nNo pending print jobs.\n");
        return;
    }

    /* Traverse queue and display details */
    temp = q->front;
    printf("\nPending Print Jobs:\n");
    printf("-------------------------------------------\n");
    printf("Job ID\tDocument Name\t\tPages\n");
    printf("-------------------------------------------\n");

    while (temp != NULL) {
        printf("%d\t%-16s\t%d\n", temp->jobId, temp->documentName, temp->pageCount);
        temp = temp->next;
    }
```

```
    printf("----------------------------------------\n");
}

/* Function to show menu options */
void displayMenu(void)
{

printf("\n========================================\n");
    printf("          PRINT QUEUE SIMULATION\n");

printf("========================================\n");
    printf("1. Add New Print Job\n");
    printf("2. Process (Print) Next Job\n");
    printf("3. Display All Pending Jobs\n");
    printf("4. Exit\n");
    printf("----------------------------------------\n");
}

/*MAIN FUNCTION*/
void main(void)
{
    struct Queue* printQueue;  /* Queue for print jobs */
    int choice;                /* User choice */
    int id, pages;             /* Job ID and page count */
    char name[100];            /* Document name */
    int running = 1;           /* Loop control variable */

    clrscr();

    printQueue = createQueue(); /* Create empty queue */

printf("========================================\n");
    printf("     Welcome to the Print Queue System      \n");
```

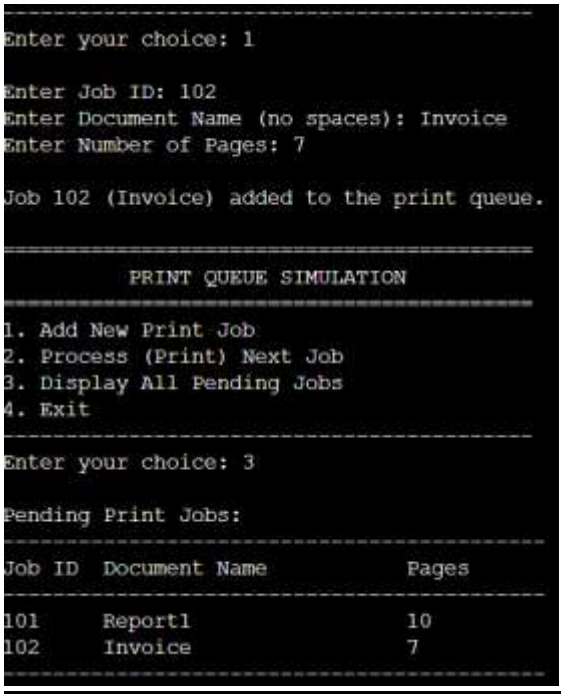
```
printf("==========================================\n");

   /* Menu-driven program loop */
   while (running) {
      displayMenu(); /* Show menu each time */
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            /* Add new job */
            printf("\nEnter Job ID: ");
            scanf("%d", &id);

            /* Input document name (no spaces for simplicity) */
            printf("Enter Document Name (no spaces): ");
            scanf("%s", name);

            /* Input number of pages */
            printf("Enter Number of Pages: ");
            scanf("%d", &pages);

            enqueue(printQueue, id, name, pages);
            break;

         case 2:
            /* Process next job */
            dequeue(printQueue);
            break;

         case 3:
            /* Display all jobs */
            display(printQueue);
            break;

         case 4:
            /* Exit program */
```

```
            printf("\nExiting Print
Queue System. Goodbye!\n");
            running = 0;
            break;

        default:
            /* Invalid input */
            printf("\nInvalid
choice! Please try again.\n");
            break;
      }
   }

   printf("\nPress any key to
exit...");
   getch();
}
```

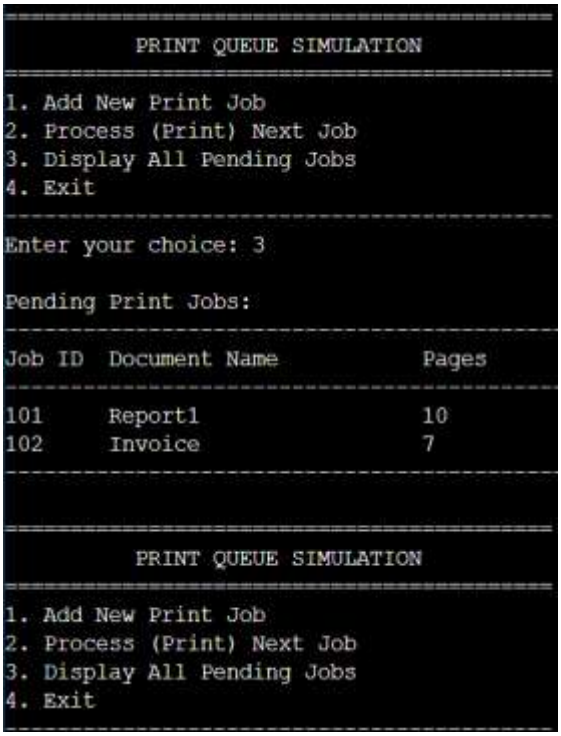
OUTPUT:



```
==========================================
     Welcome to the Print Queue System
==========================================

==========================================
          PRINT QUEUE SIMULATION
==========================================
1. Add New Print Job
2. Process (Print) Next Job
3. Display All Pending Jobs
4. Exit
------------------------------------------
Enter your choice: 1

Enter Job ID: 101
Enter Document Name (no spaces): Report1
Enter Number of Pages: 10

Job 101 (Report1) added to the print queue.

==========================================
          PRINT QUEUE SIMULATION
==========================================
1. Add New Print Job
2. Process (Print) Next Job
3. Display All Pending Jobs
4. Exit
------------------------------------------
Enter your choice:
```



```
------------------------------------------
Enter your choice: 1

Enter Job ID: 102
Enter Document Name (no spaces): Invoice
Enter Number of Pages: 7

Job 102 (Invoice) added to the print queue.

==========================================
          PRINT QUEUE SIMULATION
==========================================
1. Add New Print Job
2. Process (Print) Next Job
3. Display All Pending Jobs
4. Exit
------------------------------------------
Enter your choice: 3

Pending Print Jobs:
------------------------------------------
Job ID  Document Name           Pages
------------------------------------------
101     Report1                 10
102     Invoice                 7
------------------------------------------
```



```
==========================================
          PRINT QUEUE SIMULATION
==========================================
1. Add New Print Job
2. Process (Print) Next Job
3. Display All Pending Jobs
4. Exit
------------------------------------------
Enter your choice: 3

Pending Print Jobs:
------------------------------------------
Job ID  Document Name           Pages
------------------------------------------
101     Report1                 10
102     Invoice                 7
------------------------------------------

==========================================
          PRINT QUEUE SIMULATION
==========================================
1. Add New Print Job
2. Process (Print) Next Job
3. Display All Pending Jobs
4. Exit
------------------------------------------
```

```
Enter your choice: 2

Printing Job ID: 101
Document Name : Report1
Pages         : 10
Status        : Completed

==========================================
          PRINT QUEUE SIMULATION
==========================================
1. Add New Print Job
2. Process (Print) Next Job
3. Display All Pending Jobs
4. Exit
------------------------------------------
Enter your choice: 4

Exiting Print Queue System. Goodbye!

Press any key to exit...

...Program finished with exit code 255
Press ENTER to exit console.
```

Code Explanation:

Structure Definition

```
struct Node {
   int jobId;
   char documentName[100];
   int pageCount;
   struct Node* next;
};
```

Each node (job) stores:

jobId: Unique ID of the print job.

documentName: Name of the document.

pageCount: Pages to print.

next: Pointer to next job.

```
struct Queue {
   struct Node* front;
   struct Node* rear;
};
```

Maintains two pointers:

front → first job (to be printed next)

rear → last job added.

Enqueue (Add Job)

void enqueue(struct Queue* q, int id, char* name, int pages)

Creates a new node.

If queue empty → front = rear = new node.

Else → attaches at end (rear->next = newNode) and moves rear pointer.

Dequeue (Process Job)

void dequeue(struct Queue* q)

Removes the job from the front (first in line).

Displays job details.

Moves front pointer to the next job.

Frees memory.

Display Queue

void display(struct Queue* q)

Traverses from front to rear printing job info.

If queue empty → displays “No pending jobs.”

Menu + User Input

main() provides a menu-driven interface using a while(1) loop and a switch statement.

It repeatedly allows users to:

Add, process, display, or exit.

Key Features

Dynamic (no fixed size like arrays)

Realistic job info (ID, name, pages)

Menu-driven and user-friendly

Proper memory management (malloc / free)