Program 1:

Code:

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

typedef struct Node {

	int data;
	struct Node *next;

}Node;

Node *head = NULL;

int countNodes() {

	Node *tmp=head;
	int cnt=0;

	while(tmp != NULL) {

		cnt++;
		tmp=tmp->next;
	}

	return cnt;
}

Node *createNode() {

	Node *newNode = (Node*)malloc(sizeof(Node));

	printf("Enter data:");
	scanf("%d",&newNode->data);

	newNode->next = NULL;

	return newNode;
}

void addNode(Node *newNode) {

	if(head == NULL)
		head = newNode;
	else {

		Node *tmp = head;

		while(tmp->next != NULL)
			tmp = tmp->next;

		tmp->next = newNode;

	}
}

void addFirst(Node *newNode) {

	if(head != NULL)
```

```
                newNode->next = head;

        head = newNode;
}


void addAtPos(int pos ,Node *newNode) {


        if(pos == 1)
                addFirst(newNode);
        else if(pos ==  countNodes()+1 ) {
                addNode(newNode);
        }else {
                Node *tmp = head;

                while(pos-2) {

                        tmp=tmp->next;
                        pos--;
                }

                newNode->next = tmp->next;
                tmp->next = newNode;
        }
}

void add() {

        Node *newNode = createNode();
        int pos = 1;

        if(head == NULL) {

                addFirst(newNode);
        }else {

                Node *tmp = head;
                while(tmp != NULL) {

                        if(tmp->data <= newNode->data){

                                tmp = tmp->next;
                                pos++;
                        }else
                                break;
                }

                addAtPos(pos,newNode);
        }
}

void printList() {

        Node *tmp=head;

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

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp=tmp->next;
        }
```

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

void deleteFirst() {

        if(head == NULL) {

                printf("\nError:Linked List is already empty\n");
        }else {

                Node *tmp = head;
                head = tmp->next;
                free(tmp);

        }
}

void main() {

        int ch;

        while(1) {

                printf("\n1.Add\n");
                printf("2.Delete\n");
                printf("3.PrintList\n");
                printf("4.Exit\n");

                printf("\n Select any option from above:");
                scanf("%d",&ch);


                switch(ch) {

                        case 1:
                                add();
                                break;
                        case 2:
                                deleteFirst();
                                break;
                        case 3:
                                printList();
                                break;
                        case 4:
                                exit(0);
                                break;
                }
        }

}
```

Output:

```
sandy@sandys-Machine:~/Desktop/Study/bootcamp/DS/DailyC

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:10

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:20

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:30

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|10|->|20|->|30|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit
```

```
 Select any option from above:1
Enter data:15

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|10|->|15|->|20|->|30|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:0

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|0|->|10|->|15|->|20|->|30|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2
```

```
1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|10|->|15|->|20|->|30|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|15|->|20|->|30|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2
```

```
1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|30|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

Error:Linked List is already empty

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:4
```

Program 2:

Code:

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

typedef struct Node {

	int data;
	int priority;
	struct Node *next;

}Node;

Node *head = NULL;

int countNodes() {

	Node *tmp=head;
	int cnt=0;

	while(tmp != NULL) {

		cnt++;
		tmp=tmp->next;
	}

	return cnt;
}

Node *createNode() {

	Node *newNode = (Node*)malloc(sizeof(Node));

	printf("Enter data:");
	scanf("%d",&newNode->data);
	printf("Enter priority:");
	scanf("%d",&newNode->priority);

	if(newNode->priority < 0 || newNode->priority>5 ) {

		printf("Entered Invalid priority!..");
		printf("Enter data again\n");
		free(newNode);			//freeing node cause we are going to call createNode()
		createNode();
	}else {
		newNode->next = NULL;
		return newNode;
	}
}

void addNode(Node *newNode) {

	if(head == NULL)
		head = newNode;
	else {

		Node *tmp = head;

		while(tmp->next != NULL)
			tmp = tmp->next;
```

```
                tmp->next = newNode;

        }
}

void addFirst(Node *newNode) {

        if(head != NULL)
                newNode->next = head;

        head = newNode;
}


void addAtPos(int pos ,Node *newNode) {


        if(pos == 1)
                addFirst(newNode);
        else if(pos ==  countNodes()+1 ) {
                addNode(newNode);
        }else {
                Node *tmp = head;

                while(pos-2) {

                        tmp=tmp->next;
                        pos--;
                }

                newNode->next = tmp->next;
                tmp->next = newNode;
        }
}

void add() {

        Node *newNode = createNode();
        int pos = 1;

        if(head == NULL) {

                addFirst(newNode);
        }else {

                Node *tmp = head;
                while(tmp != NULL) {

                        if(tmp->priority <= newNode->priority){

                                tmp = tmp->next;
                                pos++;
                        }else
                                break;
                }

                addAtPos(pos,newNode);
        }
}

void printList() {
```

```
	Node *tmp=head;

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

	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp=tmp->next;
	}

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

void deleteFirst() {

	if(head == NULL) {

		printf("\nError:Linked List is already empty\n");
	}else {

		Node *tmp = head;
		head = tmp->next;
		free(tmp);

	}
}

void main() {

	int ch;

	while(1) {

		printf("\n1.Add\n");
		printf("2.Delete\n");
		printf("3.PrintList\n");
		printf("4.Exit\n");

		printf("\n Select any option from above:");
		scanf("%d",&ch);


		switch(ch) {

			case 1:
				add();
				break;
			case 2:
				deleteFirst();
				break;
			case 3:
				printList();
				break;
			case 4:
				exit(0);
				break;
		}
	}

}
```

Output:

```
1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:10
Enter priority:5

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|10|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:20
Enter priority:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|20|->|10|->NULL
-----------------------------------------------------
```

```
1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:30
Enter priority:0

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|30|->|20|->|10|->NULL
-----------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:1
Enter data:5
Enter priority:-3
Entered Invalid priority!..Enter data again
Enter data:5
Enter priority:0

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
|30|->|5|->|20|->|10|->NULL
-----------------------------------------------------
```

```
 Select any option from above:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

Error:Linked List is already empty

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:3
-----------------------------------------------------
NULL
-----------------------------------------------------
```

Program 3: Priority queue using linked list

Code:

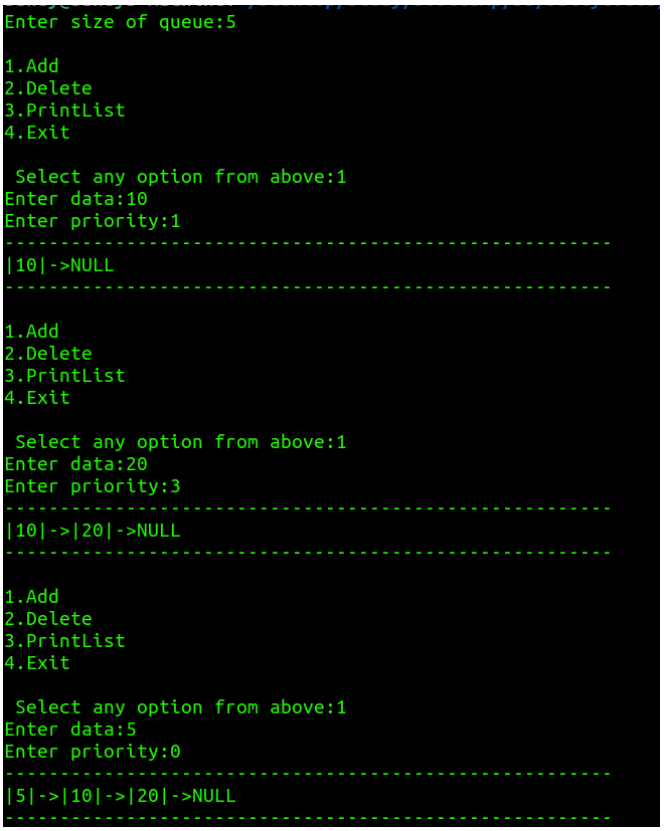
```
#include<stdio.h>
#include<stdlib.h>

typedef struct Node {

	int data;
	int priority;
	struct Node *next;

}Node;

Node *front = NULL,*rear=NULL;

int cnt=0,size,flg=0;

Node *createNode() {

	Node *newNode = (Node*)malloc(sizeof(Node));

	printf("Enter data:");
	scanf("%d",&newNode->data);
	printf("Enter priority:");
	scanf("%d",&newNode->priority);

	if(newNode->priority < 0 || newNode->priority>5 ) {

		printf("Entered Invalid priority!..");
		printf("Enter data again\n");
		free(newNode);			//freeing node cause we are going to call createNode()
		createNode();
	}else {
		newNode->next = NULL;
		return newNode;
	}
}

void addNode(Node *newNode) {

	if(front == NULL)
		front = rear = newNode;
	else {
		rear->next = newNode;
		rear = newNode;
	}
}

void addFirst(Node *newNode) {

	if(front == NULL)
		front = rear = newNode;
	else {
		newNode->next = front;
		front = newNode;
	}
}
```

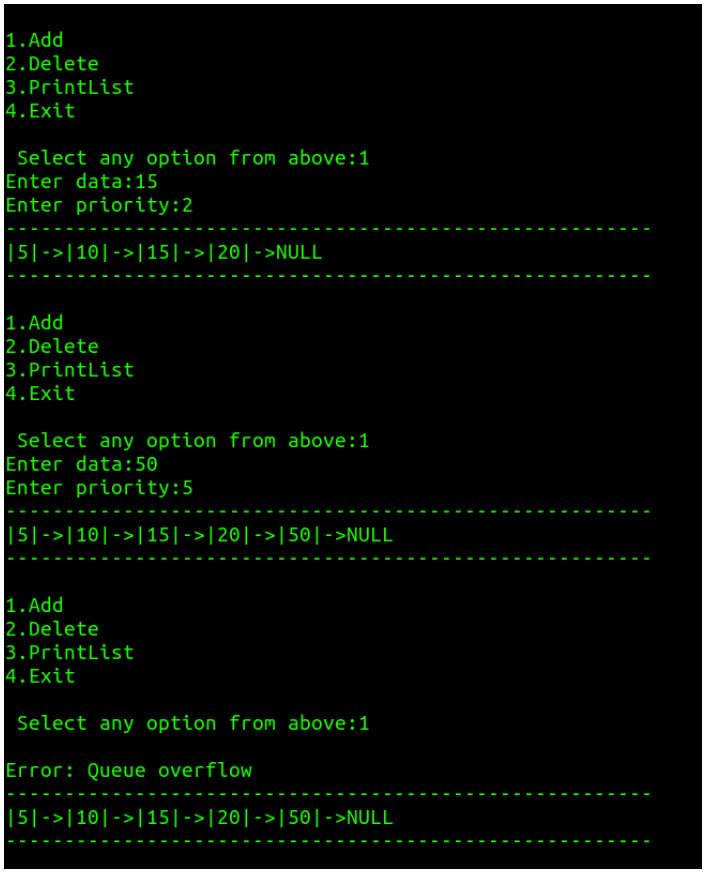
```
void addAtPos(int pos ,Node *newNode) {


	if(pos == 1)
		addFirst(newNode);
	else if(pos ==  cnt+1 ) {
		addNode(newNode);
	}else {
		Node *tmp = front;

		while(pos-2) {

			tmp=tmp->next;
			pos--;
		}

		newNode->next = tmp->next;
		tmp->next = newNode;
	}
}

int enqueue() {

	if(cnt == size) {

		return -1;
	}else {


		cnt++;
		Node *newNode = createNode();
		int pos = 1;

		if(front == NULL) {

			addFirst(newNode);
		}else {

			Node *tmp = front;
			while(tmp != NULL) {

				if(tmp->priority <= newNode->priority){

					tmp = tmp->next;
					pos++;
				}else
					break;
			}

			addAtPos(pos,newNode);
		}

		return 0;
	}
}

void printQueue() {

	Node *tmp=front;

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

	while(tmp != NULL) {
```

```
                printf("|%d|->",tmp->data);
                tmp=tmp->next;
        }

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

int dequeue() {

        if(front == NULL) {

                flg=0;
                return -1;
        }else {

                cnt--;
                flg=1;
                int data = front->data;
                Node *tmp = front;
                front = tmp->next;
                free(tmp);
                return data;
        }
}

void main() {

        printf("Enter size of queue:");
        scanf("%d",&size);

        int ch;

        while(1) {

                printf("\n1.Add\n");
                printf("2.Delete\n");
                printf("3.PrintList\n");
                printf("4.Exit\n");

                printf("\n Select any option from above:");
                scanf("%d",&ch);


                switch(ch) {

                        case 1:{
                                int ret = enqueue();

                                if(ret == -1)
                                        printf("\nError: Queue overflow\n");

                              printQueue();
                             }
                                break;
                        case 2:{
                                int ret = dequeue();

                                if(flg == 0)
                                        printf("\nError: Queue underflow\n");
                              else
                                        printf("\n%d is removed fronm queuew\n",ret);
```

```
                    printQueue();
                    }
                        break;
                case 3:
                        printQueue();
                        break;
                case 4:
                        exit(0);
                        break;
            }
        }

    }
```

Output:





```
1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

5 is removed fronm queuew
-------------------------------------------------------
|10|->|15|->|20|->|50|->NULL
-------------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

10 is removed fronm queuew
-------------------------------------------------------
|15|->|20|->|50|->NULL
-------------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

15 is removed fronm queuew
-------------------------------------------------------
|20|->|50|->NULL
-------------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit
```

```
 Select any option from above:2

20 is removed fronm queuew
-------------------------------------------------------
|50|->NULL
-------------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

50 is removed fronm queuew
-------------------------------------------------------
NULL
-------------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:2

Error: Queue underflow
-------------------------------------------------------
NULL
-------------------------------------------------------

1.Add
2.Delete
3.PrintList
4.Exit

 Select any option from above:4
```