2347214
Aravind Nandakumar
MCA - B
Christ University

Lab Exercise 4

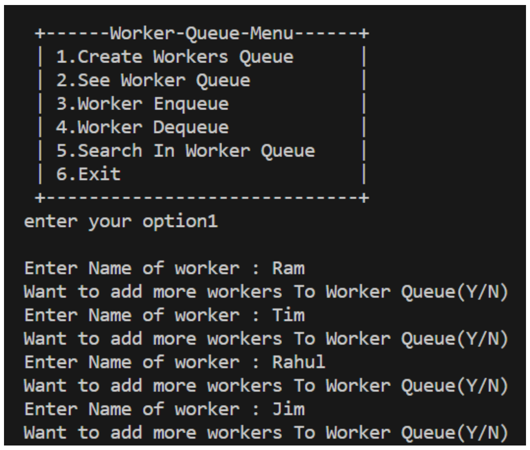
<u>Data Structures and Algorithms</u>

Implementing Queue Using Linked List

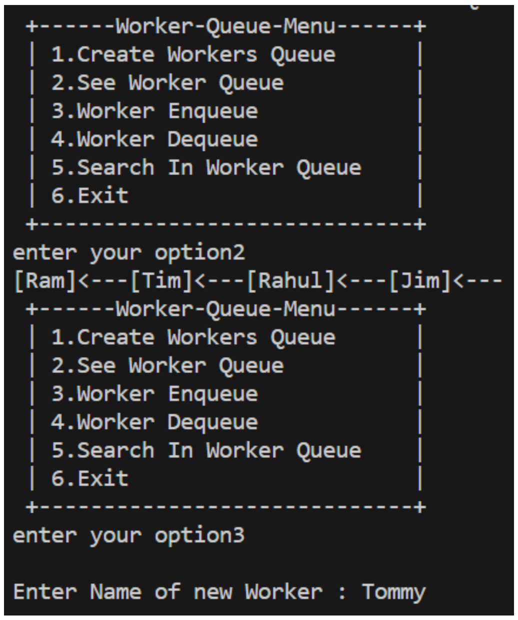
```c
C/C++
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>

struct worker{
  char name[10];
  struct worker *right;
}*newptr,*first,*last,*temp,*prev,*next;

int create(){
  char ch;
       while(1)
       {
              newptr=(struct worker*) malloc(sizeof(struct worker));
              if(newptr==NULL){
                            printf("Memory allocation error!!");
                            return 0;
                     }
              printf("\nEnter Name of worker : ");
              scanf("%s",&newptr->name);
              newptr->right=NULL;
              if(first==NULL)
                first=temp=newptr;
                else
                     {
                            temp->right=newptr;
                            temp=temp->right;
                     }
              printf("Want to add more workers To Worker Queue(Y/N)");
              ch=getch();
              if(ch=='n'||ch=='N')
```

```
    return(0);
       temp=first;
       while(temp->right!=NULL)
       {
              temp=temp->right;
              last=temp;
       }
 }
}


void peekWorkerQueue(){
  temp= first;
  if(temp==NULL){
    printf("There are no workers in Queue!!\n");
    return;
  }
  while(temp != NULL) {
    printf("[%s]<---",temp -> name );
    temp = temp-> right ;
  }

}


void workerEnqueue(){
       newptr=(struct worker*)malloc(sizeof(struct worker));
       if(newptr==NULL){
              printf("Memory allocation error!!");
              return;
       }
       printf("\nEnter Name of new Worker : ");
       scanf("%s",&newptr->name);
       newptr->right=NULL;
       temp= first;
       while(temp!=NULL){
              last=temp;
              temp=temp->right;
       }
       last->right=newptr;
       newptr->right=NULL;

}

void workerDequeue(){
```

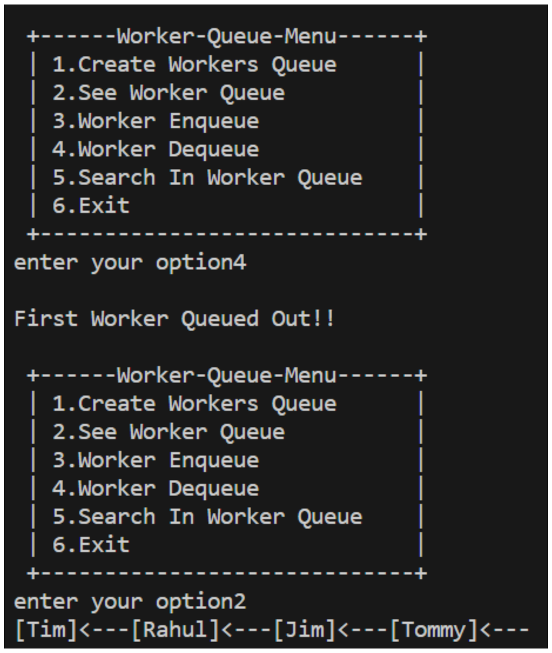
```c
        if(first==NULL){
                printf("\nThere are no Workers in Queue!!");
        }
        else{
                temp = first;
                first = first -> right;
                free(temp);
                printf("\nFirst Worker Queued Out!!\n");
        }
}

void searchInQueue(){
        char search_name[25];
        int pos ,foundFlag=0;pos=0;
        temp = first;
        printf("\nEnter name of the worker you want to find : ");
        scanf("%s",&search_name);
        while(temp!=NULL){
                pos++;
                if(strcmpi(search_name,temp->name)==0){
                        foundFlag =1;
                        printf("\nWorker Found at Position %d",pos);
                }
                temp= temp->right;
        }
        if(foundFlag==0){
                printf("\n\tNo such worker found In Queue!!");
        }
}


void exit_program(){
  temp = first;
  while (temp != NULL) {
    struct worker* nextNode = temp->right;
    free(temp);
    temp = nextNode;
  }
        exit(0);
}


void main()
{
        int opt;
        opt=0;
        first=temp=NULL;
        while(1)
   {
    printf("\n");
        printf(" +------Worker-Queue-Menu------+\n");
        printf(" | 1.Create Workers Queue      |\n");
        printf(" | 2.See Worker Queue          |\n");
        printf(" | 3.Worker Enqueue            |\n");
        printf(" | 4.Worker Dequeue            |\n");
        printf(" | 5.Search In Worker Queue    |\n");
        printf(" | 6.Exit                      |\n");
        printf(" +----------------------------+\n");
        printf("enter your option");
        scanf("%d",&opt);
        switch(opt)
        {
              case 1:create();break;
              case 2:peekWorkerQueue();break;
              case 3:workerEnqueue();break;
              case 4:workerDequeue();break;
              case 5:searchInQueue();break;
              case 6:exit_program();
        }
        getch();
     }
  }
```



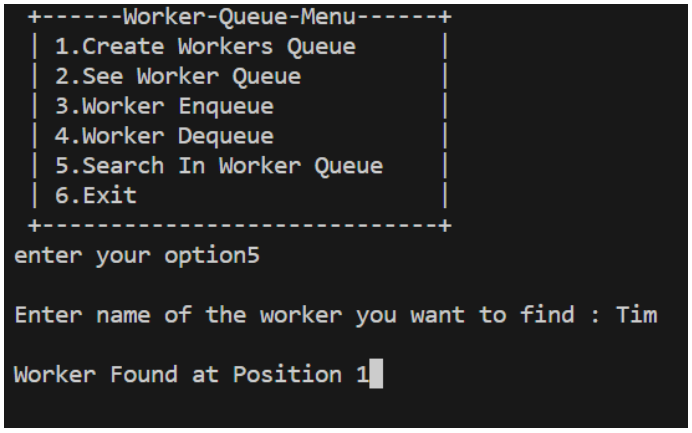
```
 +------Worker-Queue-Menu------+
 | 1.Create Workers Queue      |
 | 2.See Worker Queue          |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +----------------------------+
enter your option1

Enter Name of worker : Ram
Want to add more workers To Worker Queue(Y/N)
Enter Name of worker : Tim
Want to add more workers To Worker Queue(Y/N)
Enter Name of worker : Rahul
Want to add more workers To Worker Queue(Y/N)
Enter Name of worker : Jim
Want to add more workers To Worker Queue(Y/N)
```



```
 +------Worker-Queue-Menu------+
 | 1.Create Workers Queue      |
 | 2.See Worker Queue          |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +----------------------------+
enter your option2
[Ram]<---[Tim]<---[Rahul]<---[Jim]<---
 +------Worker-Queue-Menu------+
 | 1.Create Workers Queue      |
 | 2.See Worker Queue          |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +----------------------------+
enter your option3

Enter Name of new Worker : Tommy
```



```
 +------Worker-Queue-Menu------+
 | 1.Create Workers Queue      |
 | 2.See Worker Queue          |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +-----------------------------+
enter your option4

First Worker Queued Out!!

 +------Worker-Queue-Menu------+
 | 1.Create Workers Queue      |
 | 2.See Worker Queue          |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +-----------------------------+
enter your option2
[Tim]<---[Rahul]<---[Jim]<---[Tommy]<---
```



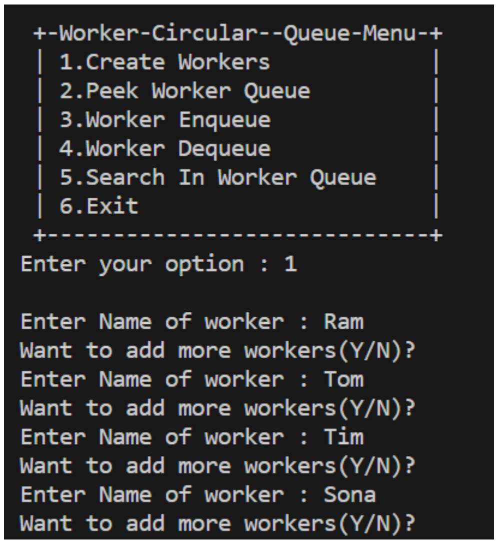
```
 +------Worker-Queue-Menu------+
 | 1.Create Workers Queue      |
 | 2.See Worker Queue          |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +-----------------------------+
enter your option5

Enter name of the worker you want to find : Tim

Worker Found at Position 1
```

Implementing Circular Queue Using Linked List

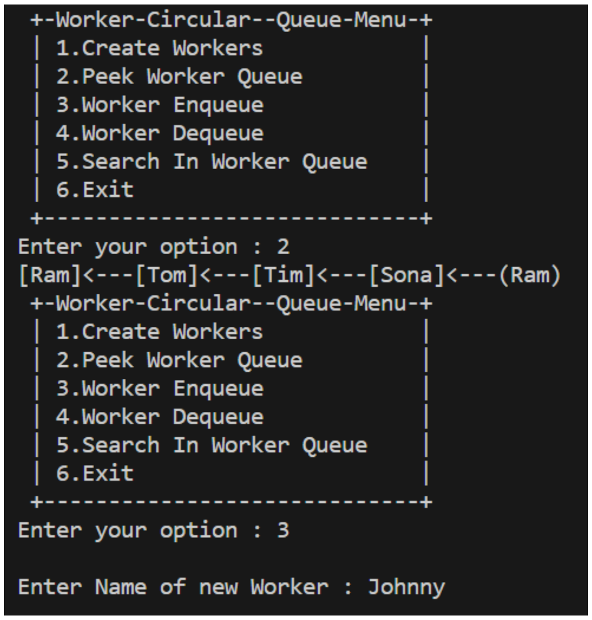
C/C++
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>

struct worker{
  char name[10];
  struct worker *right;
}*newptr,*first,*last,*temp,*prev,*next;


int create(){
  char ch;
       while(1)
       {
              newptr=(struct worker*) malloc(sizeof(struct worker));
              if(newptr==NULL){
                     printf("Memory allocation error!!");
                     return 0;
         }
              printf("\nEnter Name of worker : ");
              scanf("%s",&newptr->name);
              newptr->right=NULL;
              if(first==NULL)
               first=temp=last=newptr;
               else
                     {
                            temp->right=newptr;
                            temp=temp->right;
                     }
              printf("Want to add more workers(Y/N)?");
              ch=getch();
              if(ch=='n'||ch=='N')
    {
                     temp=first;
                     while(temp->right!=NULL)
                     {
                            temp=temp->right;
                            last=temp;
                     }
                     last->right= first;
                     return(0);
              }
 }

}


void peekWorkerQueue(){
  temp= first;
  if(temp==NULL){
    printf("There are no workers In Queue!!\n");
    return;
  }
       do{
              printf("[%s]<---",temp -> name );
    temp = temp-> right ;
       }while(temp!=first);
       printf("(%s)",last->right->name);
}

void workerEnqueue(){
       newptr=(struct worker*)malloc(sizeof(struct worker));
       if(newptr==NULL){
                     printf("Memory allocation error!!");
                     return;
         }
       printf("\nEnter Name of new Worker : ");
       scanf("%s",&newptr->name);
       newptr->right=NULL;
       last->right=newptr;
  last=newptr;
  last->right=first;

}

void workerDequeue(){
       if(first==NULL){
              printf("\nThere are no Workers in Queue");
       }
       else{
              temp = first;
              first = first -> right;
              last->right=first;
              free(temp);
              printf("\nFirst Worker Queued Out!!\n");
```

```
        }
}


void searchWorkerQueue(){
        char search_name[25];
        int pos ,foundFlag=0;pos=0;
        temp = first;
        printf("\nEnter name of the worker you want to find : ");
        scanf("%s",&search_name);

        do{
                pos++;
                if(strcmpi(search_name,temp->name)==0){
                        foundFlag =1;
                        printf("\nWorker Found at Position %d",pos);
                        break;
                }
                temp= temp->right;

        }while(temp!=first);

        if(foundFlag==0){
                printf("\n\tNo such worker found In Queue!!");
        }
}

void exit_program(){
  temp = first;
  while (temp != NULL) {
    struct worker* nextNode = temp->right;
    free(temp);
    temp = nextNode;
  }
        exit(0);
}


void main()
{
        int opt;
        opt=0;
        first=temp=NULL;
        while(1)
```

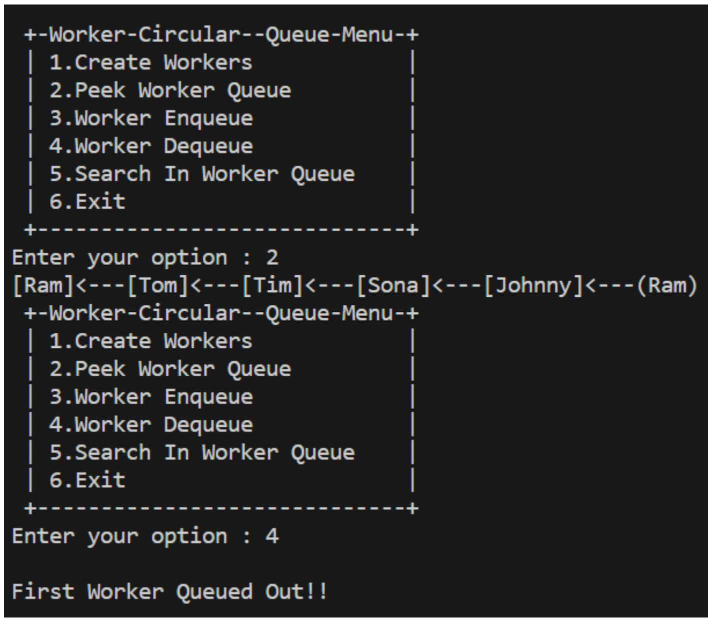
```
 {
  printf("\n");
      printf(" +-Worker-Circular--Queue-Menu-+\n");
      printf(" | 1.Create Workers            |\n");
      printf(" | 2.Peek Worker Queue         |\n");
      printf(" | 3.Worker Enqueue            |\n");
      printf(" | 4.Worker Dequeue            |\n");
      printf(" | 5.Search In Worker Queue    |\n");
      printf(" | 6.Exit                      |\n");
  printf(" +-----------------------------+\n");
      printf("Enter your option : ");
      scanf("%d",&opt);
      switch(opt)
      {
             case 1:create();break;
             case 2:peekWorkerQueue();break;
             case 3:workerEnqueue();break;
             case 4:workerDequeue();break;
             case 5:searchWorkerQueue();break;
             case 6:exit_program();
      }
      getch();
 }
}
```



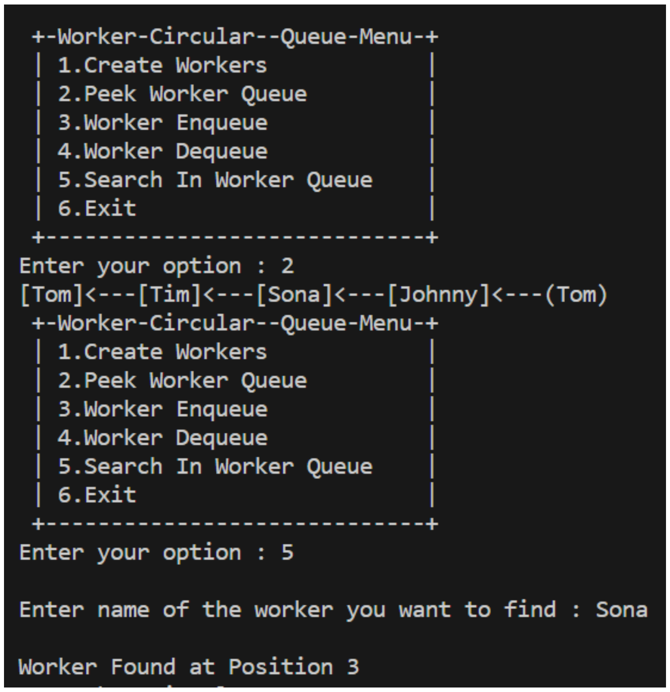
```
 +-Worker-Circular--Queue-Menu-+
 | 1.Create Workers            |
 | 2.Peek Worker Queue         |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +-----------------------------+
Enter your option : 1

Enter Name of worker : Ram
Want to add more workers(Y/N)?
Enter Name of worker : Tom
Want to add more workers(Y/N)?
Enter Name of worker : Tim
Want to add more workers(Y/N)?
Enter Name of worker : Sona
Want to add more workers(Y/N)?
```



```
 +-Worker-Circular--Queue-Menu-+
 | 1.Create Workers            |
 | 2.Peek Worker Queue         |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +-----------------------------+
Enter your option : 2
[Ram]<---[Tom]<---[Tim]<---[Sona]<---(Ram)
 +-Worker-Circular--Queue-Menu-+
 | 1.Create Workers            |
 | 2.Peek Worker Queue         |
 | 3.Worker Enqueue            |
 | 4.Worker Dequeue            |
 | 5.Search In Worker Queue    |
 | 6.Exit                      |
 +-----------------------------+
Enter your option : 3

Enter Name of new Worker : Johnny
```



```
 +-Worker-Circular--Queue-Menu-+
 | 1.Create Workers             |
 | 2.Peek Worker Queue          |
 | 3.Worker Enqueue             |
 | 4.Worker Dequeue             |
 | 5.Search In Worker Queue     |
 | 6.Exit                       |
 +-----------------------------+
Enter your option : 2
[Ram]<---[Tom]<---[Tim]<---[Sona]<---[Johnny]<---(Ram)
 +-Worker-Circular--Queue-Menu-+
 | 1.Create Workers             |
 | 2.Peek Worker Queue          |
 | 3.Worker Enqueue             |
 | 4.Worker Dequeue             |
 | 5.Search In Worker Queue     |
 | 6.Exit                       |
 +-----------------------------+
Enter your option : 4

First Worker Queued Out!!
```



```
 +-Worker-Circular--Queue-Menu-+
 | 1.Create Workers             |
 | 2.Peek Worker Queue          |
 | 3.Worker Enqueue             |
 | 4.Worker Dequeue             |
 | 5.Search In Worker Queue     |
 | 6.Exit                       |
 +-----------------------------+
Enter your option : 2
[Tom]<---[Tim]<---[Sona]<---[Johnny]<---(Tom)
 +-Worker-Circular--Queue-Menu-+
 | 1.Create Workers             |
 | 2.Peek Worker Queue          |
 | 3.Worker Enqueue             |
 | 4.Worker Dequeue             |
 | 5.Search In Worker Queue     |
 | 6.Exit                       |
 +-----------------------------+
Enter your option : 5

Enter name of the worker you want to find : Sona

Worker Found at Position 3
```

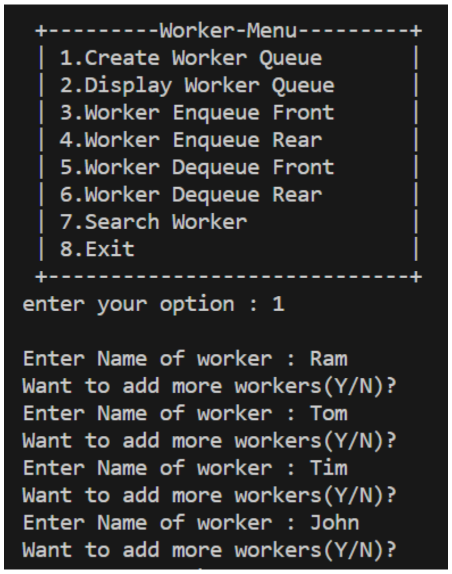
Implementing Double Ended Queue Using Linked List

C/C++

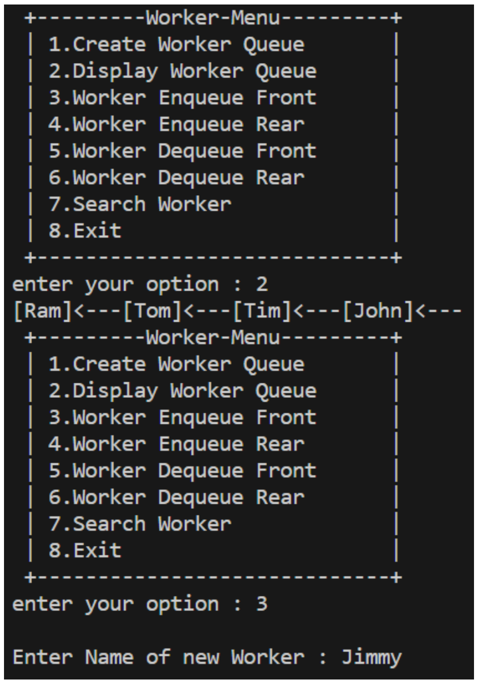
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>

struct worker{
  char name[10];
  struct worker *right;
}*newptr,*first,*last,*temp,*prev,*next;

int numberOfNodes=0;


int create(){
  char ch;
       while(1)
       {
              newptr=(struct worker*) malloc(sizeof(struct worker));
              if(newptr==NULL){
                            printf("Memory allocation error!!");
                            return 0;
                     }
              printf("\nEnter Name of worker : ");
              scanf("%s",&newptr->name);
              numberOfNodes++;
              newptr->right=NULL;
              if(first==NULL)
                first=temp=newptr;
                else
                     {
                            temp->right=newptr;
                            temp=temp->right;
                     }
              printf("Want to add more workers(Y/N)?");
              ch=getch();
              if(ch=='n'||ch=='N')
    return(0);
       temp=first;
       while(temp->right!=NULL)
       {
              temp=temp->right;
              last=temp;
       }
 }
}


void peekWorkerQueue(){
  temp= first;
  if(temp==NULL){
    printf("There are no workers!!\n");
    return;
  }
  while(temp != NULL) {
    printf("[%s]<---",temp -> name );
    temp = temp-> right ;
  }
}


void enqueueFront(){
       newptr = (struct worker *)malloc( sizeof( struct worker ) );
       if(newptr==NULL){
              printf("Memory allocation error");
              return;
       }
       printf("\nEnter Name of new Worker : ");
       scanf("%s",&newptr->name);
       numberOfNodes++;
       newptr->right=NULL;
       if(first == NULL)
       {
              first=last=newptr;
  }
       else
       {
              newptr->right=first;
              first=newptr;
       }
}

void enqueueRear(){
       newptr=(struct worker*)malloc(sizeof(struct worker));
       if(newptr==NULL){
              printf("Memory allocation error");
              return;
       }
```

```
        printf("\nEnter Name of new Worker : ");
        scanf("%s",&newptr->name);
        numberOfNodes++;
        newptr->right=NULL;
        temp= first;
        while(temp!=NULL){
                last=temp;
                temp=temp->right;
        }
        last->right=newptr;
        newptr->right=NULL;
}


void dequeueFront(){
        if(first==NULL){
                printf("\nThere are no Workers");
        }
        else{
                temp = first;
                first = first -> right;
                free(temp);
                numberOfNodes--;
                printf("\nFirst Worker deleted\n");


        }
}

void dequeueRear(){
        if(first==NULL){
                printf("\nThere are no Workers !!");
                return;
        }
        temp=first;
        while(temp->right!=NULL)
        {
                prev=temp;
                temp=temp->right;
                last=temp;
        }
        prev->right=NULL;
        last=prev;
  printf("\nLast Worker Queued Out!!\n");
        numberOfNodes--;
```

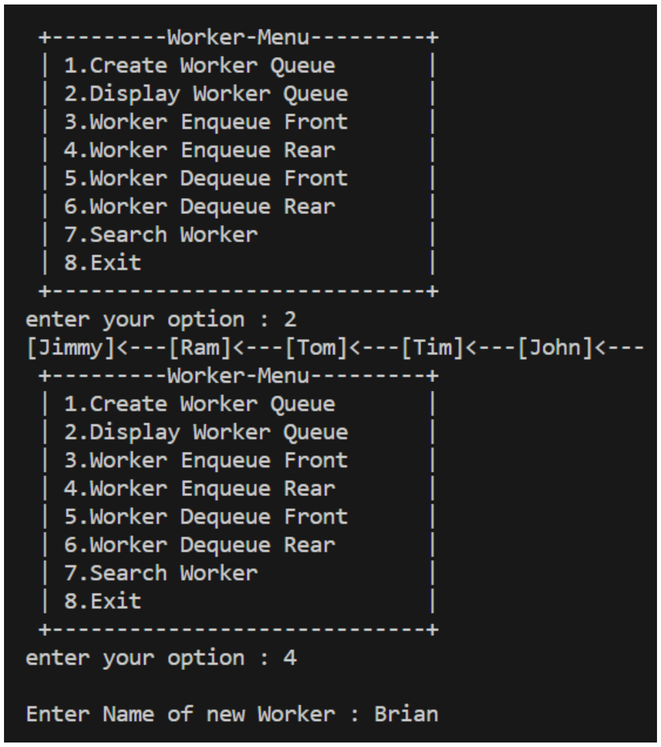
```c
        free(temp);
}

void searchWorkerQueue(){
        char search_name[25];
        int pos ,foundFlag=0;pos=0;
        temp = first;
        printf("\nEnter name of the worker you want to find : ");
        scanf("%s",&search_name);
        while(temp!=NULL){
                pos++;
                if(strcmpi(search_name,temp->name)==0){
                        foundFlag =1;
                        printf("\nWorker Found at %d",pos);
                }
                temp= temp->right;
        }
        if(foundFlag==0){
                printf("\n\tNo such worker found!");
        }
}

void exit_program(){
  temp = first;
  while (temp != NULL) {
    struct worker* nextNode = temp->right;
    free(temp);
    temp = nextNode;
  }
        exit(0);
}


void main()
{
        int opt;
        opt=0;
        first=temp=NULL;
        while(1)
 {
  printf("\n");
        printf(" +---------Worker-Menu---------+\n");
        printf(" | 1.Create Worker Queue       |\n");
        printf(" | 2.Display Worker Queue      |\n");
        printf(" | 3.Worker Enqueue Front      |\n");
        printf(" | 4.Worker Enqueue Rear       |\n");
        printf(" | 5.Worker Dequeue Front      |\n");
        printf(" | 6.Worker Dequeue Rear       |\n");
        printf(" | 7.Search Worker             |\n");
        printf(" | 8.Exit                      |\n");
    printf(" +---------------------------+\n");
        printf("enter your option : ");
        scanf("%d",&opt);
        switch(opt)
        {
              case 1:create();break;
      case 2:peekWorkerQueue();break;
              case 3:enqueueFront();break;
              case 4:enqueueRear();break;
              case 5:dequeueFront();break;
              case 6:dequeueRear();break;
              case 7:searchWorkerQueue();break;
              case 8:exit_program();
        }
        getch();
     }
  }
```



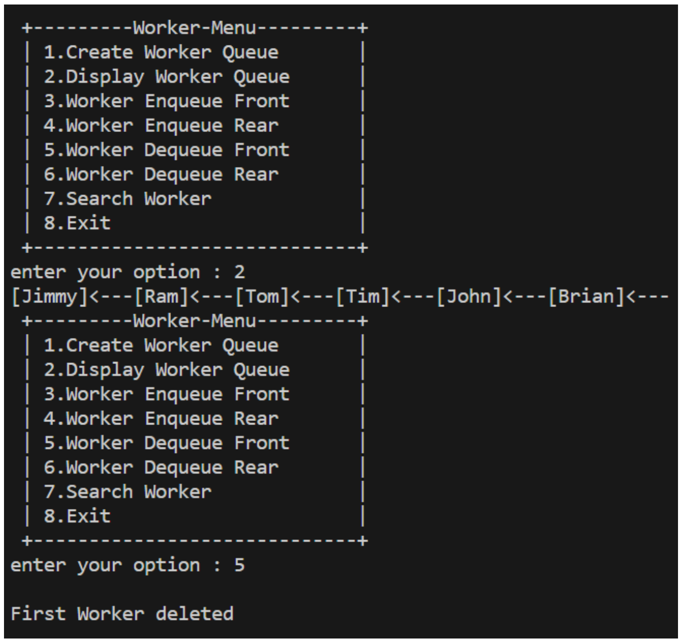
```
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +---------------------------+
enter your option : 1

Enter Name of worker : Ram
Want to add more workers(Y/N)?
Enter Name of worker : Tom
Want to add more workers(Y/N)?
Enter Name of worker : Tim
Want to add more workers(Y/N)?
Enter Name of worker : John
Want to add more workers(Y/N)?
```



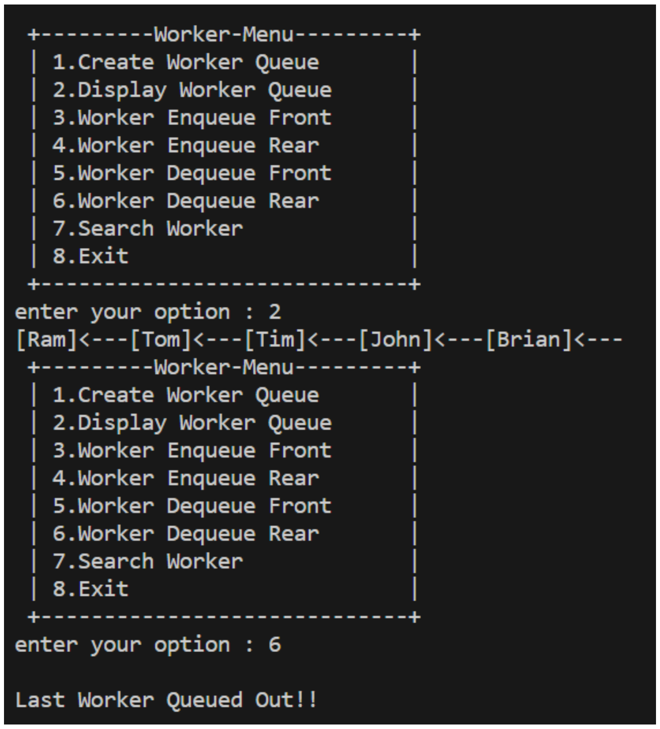
```
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +---------------------------+
enter your option : 2
[Ram]<---[Tom]<---[Tim]<---[John]<---
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +---------------------------+
enter your option : 3

Enter Name of new Worker : Jimmy
```



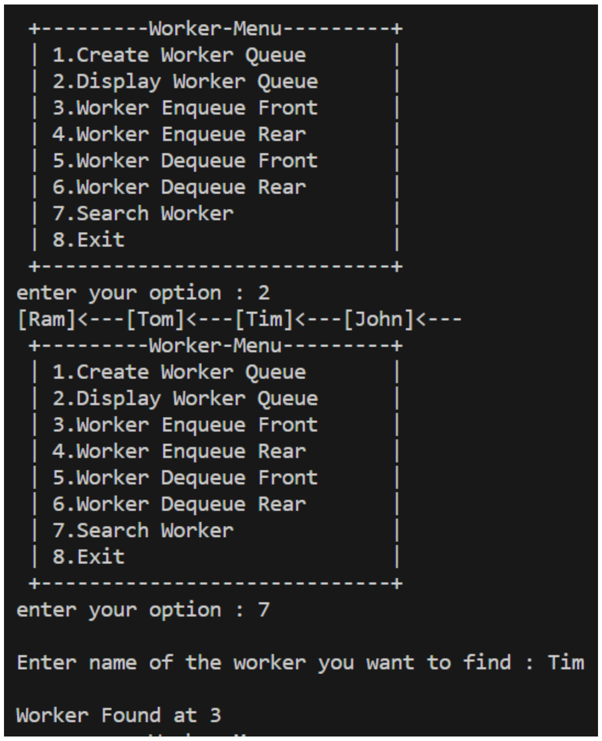
```
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 2
[Jimmy]<---[Ram]<---[Tom]<---[Tim]<---[John]<---
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 4

Enter Name of new Worker : Brian
```



```
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 2
[Jimmy]<---[Ram]<---[Tom]<---[Tim]<---[John]<---[Brian]<---
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 5

First Worker deleted
```



```
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 2
[Ram]<---[Tom]<---[Tim]<---[John]<---[Brian]<---
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 6

Last Worker Queued Out!!
```



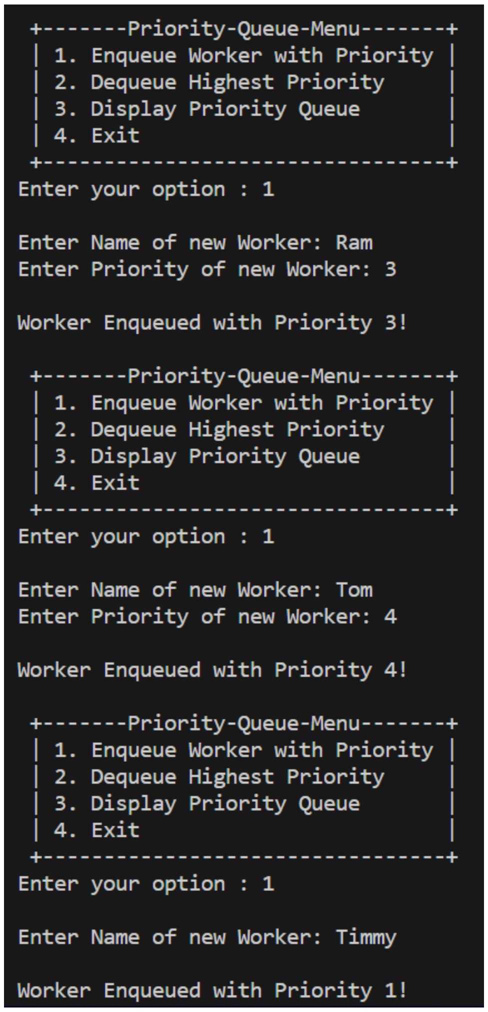
```
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 2
[Ram]<---[Tom]<---[Tim]<---[John]<---
 +---------Worker-Menu---------+
 | 1.Create Worker Queue       |
 | 2.Display Worker Queue      |
 | 3.Worker Enqueue Front      |
 | 4.Worker Enqueue Rear       |
 | 5.Worker Dequeue Front      |
 | 6.Worker Dequeue Rear       |
 | 7.Search Worker             |
 | 8.Exit                      |
 +-----------------------------+
enter your option : 7

Enter name of the worker you want to find : Tim

Worker Found at 3
```

Implementing Priority Queue Using Linked List

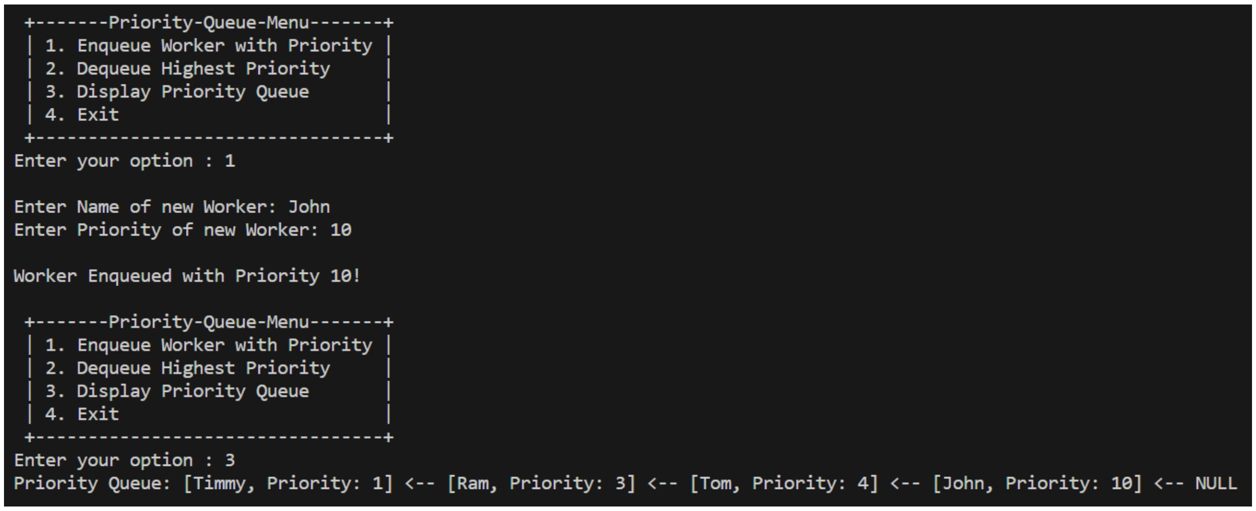
C/C++

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

struct worker {
  char name[10];
  int priority;
  struct worker* next;
};

struct worker* front = NULL;

void workerEnqueue() {
  struct worker* newptr = (struct worker*)malloc(sizeof(struct worker));
  if (newptr == NULL) {
    printf("Memory allocation error!!");
    return;
  }

  printf("\nEnter Name of new Worker: ");
  scanf("%s", newptr->name);

  printf("Enter Priority of new Worker: ");
  scanf("%d", &newptr->priority);

  newptr->next = NULL;

  if (front == NULL || newptr->priority < front->priority) {
    newptr->next = front;
    front = newptr;
  } else {
    struct worker* temp = front;
    while (temp->next != NULL && temp->next->priority <= newptr->priority) {
      temp = temp->next;
    }
    newptr->next = temp->next;
    temp->next = newptr;
  }

  printf("\nWorker Enqueued with Priority %d!\n", newptr->priority);
}
```

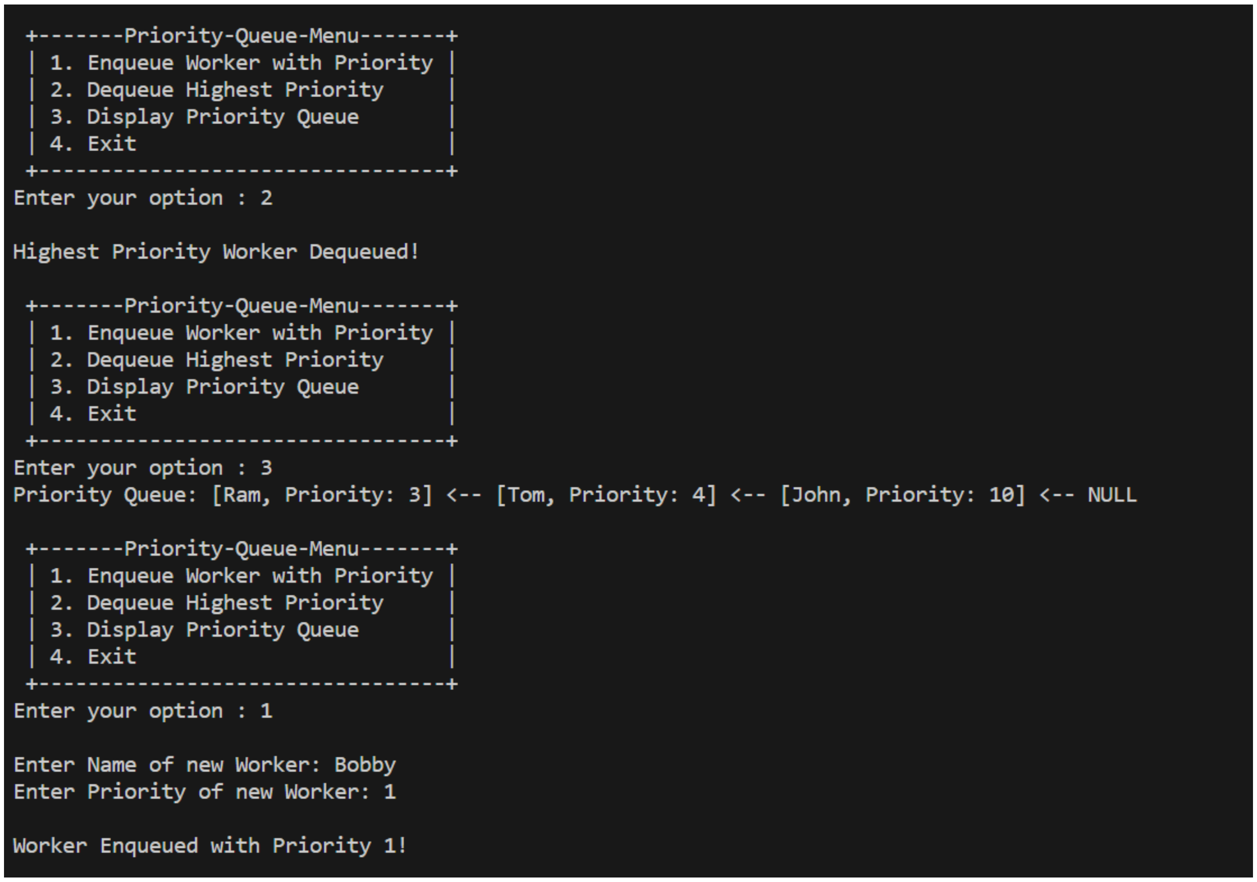
```c
void workerDequeue() {
  if (front == NULL) {
    printf("\nThere are no Workers in the Priority Queue!\n");
  } else {
    struct worker* temp = front;
    front = front->next;
    free(temp);
    printf("\nHighest Priority Worker Dequeued!\n");
  }
}

void peekPriorityQueue() {
  struct worker* temp = front;

  if (temp == NULL) {
    printf("Priority Queue is Empty!\n");
    return;
  }

  printf("Priority Queue: ");
  while (temp != NULL) {
    printf("[%s, Priority: %d] <-- ", temp->name, temp->priority);
    temp = temp->next;
  }
  printf("NULL\n");
}

int main() {
  int opt;

  while (1) {
    printf("\n");
    printf(" +-------Priority-Queue-Menu-------+\n");
    printf(" | 1. Enqueue Worker with Priority |\n");
    printf(" | 2. Dequeue Highest Priority     |\n");
    printf(" | 3. Display Priority Queue       |\n");
    printf(" | 4. Exit                         |\n");
    printf(" +---------------------------------+\n");
    printf("Enter your option : ");
    scanf("%d", &opt);

    switch (opt) {
      case 1:
        workerEnqueue();
        break;
      case 2:
        workerDequeue();
        break;
      case 3:
        peekPriorityQueue();
        break;
      case 4:
        exit(0);
      default:
        printf("Invalid Option! Please try again.\n");
    }
  }

  return 0;
}
```



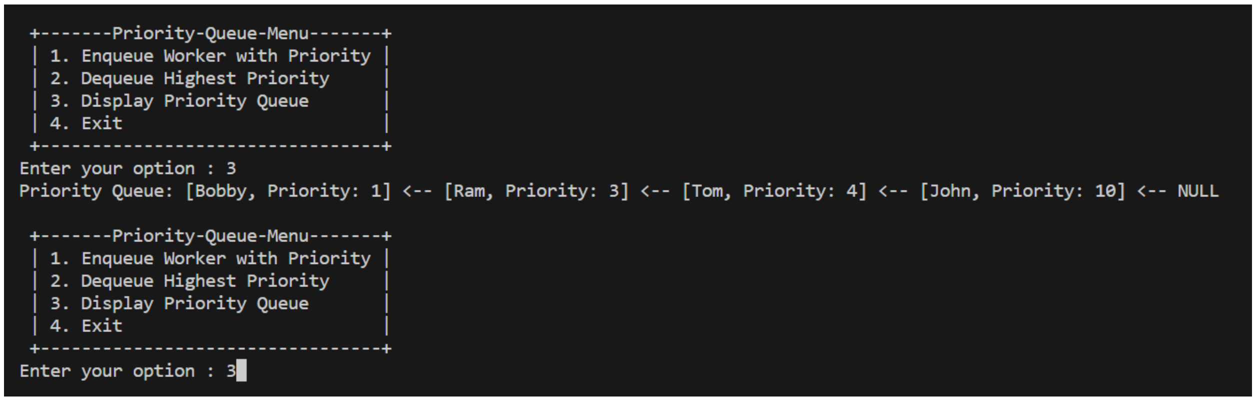
```
 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 1

Enter Name of new Worker: Ram
Enter Priority of new Worker: 3

Worker Enqueued with Priority 3!

 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 1

Enter Name of new Worker: Tom
Enter Priority of new Worker: 4

Worker Enqueued with Priority 4!

 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 1

Enter Name of new Worker: Timmy

Worker Enqueued with Priority 1!
```



```
 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 1

Enter Name of new Worker: John
Enter Priority of new Worker: 10

Worker Enqueued with Priority 10!

 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 3
Priority Queue: [Timmy, Priority: 1] <-- [Ram, Priority: 3] <-- [Tom, Priority: 4] <-- [John, Priority: 10] <-- NULL
```



```
 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 2

Highest Priority Worker Dequeued!

 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 3
Priority Queue: [Ram, Priority: 3] <-- [Tom, Priority: 4] <-- [John, Priority: 10] <-- NULL

 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 1

Enter Name of new Worker: Bobby
Enter Priority of new Worker: 1

Worker Enqueued with Priority 1!
```



```
 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 3
Priority Queue: [Bobby, Priority: 1] <-- [Ram, Priority: 3] <-- [Tom, Priority: 4] <-- [John, Priority: 10] <-- NULL

 +-------Priority-Queue-Menu-------+
 | 1. Enqueue Worker with Priority |
 | 2. Dequeue Highest Priority     |
 | 3. Display Priority Queue       |
 | 4. Exit                         |
 +---------------------------------+
Enter your option : 3
```