**Question # 1:**

**PROGRAM**

#include<iostream>

using namespace std;

int Size;

int \*queue; // Initillizing Elements

int front = -1;

int rear = -1;

void enqueue(int value) //ENQUEUE FUNCTION

{

if (front == -1 && rear == -1)

{

front = 0;

rear = 0;

queue[rear] = value;

}

else if (rear != Size - 1)

{

rear++;

queue[rear] = value;

}

else if (rear == Size - 1)

{

cout << "Queue is Over Flown" << endl;

}

}

void dequeue() //DEQUEUE FUNCTION

{

if (front != -1 && front != rear)

{

cout << endl << "Following Value is Dequeued : " << queue[front];

front++;

}

else if (front == rear)

{

cout << endl << "Following Value is Dequeued : " << queue[front];

front = -1;

rear = -1;

}

else if (front == -1 && rear == -1)

{

cout << "Queue is Empty" << endl;

}

}

void IS\_EMPTY() //Is Empty Function

{

if (front == -1 && rear == -1)

{

cout << endl << "The Queue is EMPTY !!" << endl;

}

else

cout << endl << "The Queue is Not Empty !!" << endl;

}

void IS\_FULL() // Is Full function

{

if (rear == Size - 1)

{

cout << endl << "The Queue is FULL !!" << endl;

}

else

cout << endl << "The Queue is Not FULL !!" << endl;

}

void display() // Display Function for ease

{

int temp = front;

if (rear == -1)

{

cout << "Queue is empty" << endl;

}

else

{

while (temp != rear)

{

cout << queue[temp] << " ";

temp++;

}

}

if (temp == rear)

{

cout << queue[temp] << " ";

}

}

int main()

{

cout << "Enter the Size of the Queue : ";

cin >> Size;

queue = new int[Size];

int value;

int choice = 1;

int Opt;

while (choice != 0)

{

system("cls");

cout << "---------------------------------" << endl;

cout << " Press 1 To Enqueue" << endl;

cout << " Press 2 To Dequeue" << endl;

cout << " Press 3 To Check if Queue is FULL" << endl; //Menu Driven Program

cout << " Press 4 To Check if Queue is EMPTY" << endl;

cout << " Press 5 To Display" << endl;

cout << " Press 0 To Exit" << endl;

cout << "---------------------------------" << endl;

cout << "Enter Your Option" << endl;

cin >> Opt;

switch (Opt)

{

case 1:

{

cout << "Enter Value to Enqueue : " << endl;

cin >> value;

enqueue(value);

cout << endl;

system("pause");

break;

}

case 2:

{

dequeue();

cout << endl << endl;

system("pause");

break;

}

case 3:

{

IS\_FULL();

cout << endl << endl;

system("pause");

break;

}

case 4:

{

IS\_EMPTY();

cout << endl << endl;

system("pause");

break;

}

case 5:

{

display();

cout << endl << endl;

system("pause");

break;

}

case 0:

{

choice = 0;

break;

}

default:

{

cout << "Invalid Entry" << endl;

break;

}

}

}

cout << endl << endl;

system("pause");

}

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Question # 2:**

**PROGRAM**

#include<iostream>

using namespace std;

struct Node

{

int Value;

Node \*NextNode; // Initillizing Elements

};

class Queue

{

public:

Node \*front, \*rear;

Queue()

{

front = NULL, rear = NULL;

}

void enqueue(int value) //ENQUEUE FUNCTION

{

Node \*temp = new Node;

if (front == NULL && rear == NULL)

{

temp->Value = value;

temp->NextNode = NULL;

front = temp;

rear = temp;

}

else

{

temp->Value = value;

temp->NextNode = NULL;

rear->NextNode = temp;

rear = temp;

}

}

void dequeue() //DEQUEUE FUNCTION

{

Node \*temp = new Node;

if (front != -NULL && front != rear)

{

cout << endl << "Following Value is Dequeued : " << front->Value;

temp = front;

front=front->NextNode;

free(temp);

}

else if (front == NULL && rear == NULL)

{

cout << "Queue is Empty" << endl;

}

else if (front == rear)

{

cout << endl << "Following Value is Dequeued : " << front->Value;

temp = front;

free(temp);

front = NULL;

rear = NULL;

}

}

void IS\_EMPTY() //Is Empty Function

{

if (front == NULL && rear == NULL)

{

cout << endl << "The Queue is EMPTY !!" << endl;

}

else

cout << endl << "The Queue is Not Empty !!" << endl;

}

void display() // Display Function for ease

{

Node \*temp = front;

if (temp == NULL)

{

cout << "Queue is empty" << endl;

}

else

{

while (temp != NULL)

{

cout << temp->Value << " ";

temp = temp->NextNode;

}

}

}

};

int main()

{

Queue queue;

int value;

int choice = 1;

int Opt;

while (choice != 0)

{

system("cls");

cout << "---------------------------------" << endl;

cout << " Press 1 To Enqueue" << endl;

cout << " Press 2 To Dequeue" << endl;

cout << " Press 3 To Check if Queue is EMPTY" << endl; //Menu Driven Program

cout << " Press 4 To Display" << endl;

cout << " Press 0 To Exit" << endl;

cout << "---------------------------------" << endl;

cout << "Enter Your Option" << endl;

cin >> Opt;

switch (Opt)

{

case 1:

{

cout << "Enter Value to Enqueue : " << endl;

cin >> value;

queue.enqueue(value);

cout << endl;

system("pause");

break;

}

case 2:

{

queue.dequeue();

cout << endl << endl;

system("pause");

break;

}

case 3:

{

queue.IS\_EMPTY();

cout << endl << endl;

system("pause");

break;

}

case 4:

{

queue.display();

cout << endl << endl;

system("pause");

break;

}

case 0:

{

choice = 0;

break;

}

default:

{

cout << "Invalid Entry" << endl;

break;

}

}

}

cout << endl << endl;

system("pause");

}

A screenshot of a computer screen

Description automatically generated

**Question # 3:**

**PROGRAM**

#include<iostream>

using namespace std;

struct Node

{

int Value;

Node \*NextNode; // Initillizing Elements

};

class Queue

{

public:

Node \*front, \*rear;

Queue()

{

front = NULL, rear = NULL;

}

void enqueue(int value) //ENQUEUE FUNCTION

{

Node \*temp = new Node;

if (front == NULL && rear == NULL)

{

temp->Value = value;

temp->NextNode = NULL;

front = temp;

rear = temp;

}

else

{

temp->Value = value;

temp->NextNode = NULL;

rear->NextNode = temp;

rear = temp;

}

}

void dequeue() //DEQUEUE FUNCTION

{

Node \*temp = new Node;

if (front != -NULL && front != rear)

{

cout << endl << "Following Value is Dequeued : " << front->Value;

temp = front;

front=front->NextNode;

free(temp);

}

else if (front == NULL && rear == NULL)

{

cout << "Queue is Empty" << endl;

}

else if (front == rear)

{

cout << endl << "Following Value is Dequeued : " << front->Value;

temp = front;

free(temp);

front = NULL;

rear = NULL;

}

}

void display() // Display Function for ease

{

Node \*temp = front;

if (temp == NULL)

{

cout << "Queue is empty" << endl;

}

else

{

while (temp != NULL)

{

cout << temp->Value << " ";

temp = temp->NextNode;

}

}

}

};

int main()

{

Queue queue;

int value;

int choice = 1;

int Opt;

while (choice != 0)

{

system("cls");

cout << "---------------------------------" << endl;

cout << " Press 1 To Enqueue" << endl;

cout << " Press 2 To Dequeue" << endl;

cout << " Press 3 To Display" << endl; //Menu Driven Program

cout << " Press 0 To Exit" << endl;

cout << "---------------------------------" << endl;

cout << "Enter Your Option" << endl;

cin >> Opt;

switch (Opt)

{

case 1:

{

cout << "Enter Value to Enqueue : " << endl;

cin >> value;

for (int i = 1; i <= value; i++)

{ // make nodes according to given value

queue.enqueue(value);

}

cout << endl;

system("pause");

break;

}

case 2:

{

queue.dequeue();

cout << endl << endl;

system("pause");

break;

}

case 3:

{

queue.display();

cout << endl << endl;

system("pause");

break;

}

case 0:

{

choice = 0;

break;

}

default:

{

cout << "Invalid Entry" << endl;

break;

}

}

}

cout << endl << endl;

system("pause");

}

A screenshot of a computer screen

Description automatically generated

**Question # 5:**

**PROGRAM**

#include<iostream>

#include<Windows.h>

using namespace std;

struct Node

{

int processors,time;

Node \*NextNode; // Initillizing Elements

};

class Queue

{

public:

Node \*front, \*rear;

Queue()

{

front = NULL, rear = NULL;

}

void enqueue(int proc, int tim) //ENQUEUE FUNCTION

{

Node \*temp = new Node;

if (front == NULL && rear == NULL)

{

temp->processors = proc;

temp->time = tim;

temp->NextNode = NULL;

front = temp;

rear = temp;

}

else

{

temp->processors = proc;

temp->time = tim;

temp->NextNode = NULL;

rear->NextNode = temp;

rear = temp;

}

}

void Processing()

{

if (front == NULL && rear == NULL)

{

cout << endl << "Queue is Empty !!" << endl;

}

else

{

while (front != NULL)

{

cout << front->processors << " Processor/s Takes " << front->time <<" Seconds"<< endl;

Sleep(front->time \* 1000); // Coverting Milli Second to Second

front = front->NextNode;

}

}

}

};

int main()

{

Queue queue;

int Processors = 0, Time\_Duration = 0, Sentinal = 0;

while (Sentinal != 1)

{

system("cls");

cout << "Enter The Number Of Processors : ";

cin >> Processors;

cout << "Enter The Time Duration For Given Processors : ";

cin >> Time\_Duration;

queue.enqueue(Processors, Time\_Duration);

cout << endl << "Enter 1 to Check Time Duration of Provided Processors " << endl;

cout << " OR " << endl;

cout << "Enter Any Number to Repeat the Same Process : ";

cin >> Sentinal;

}

cout << endl;

queue.Processing();

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**Question # 6:**

**PROGRAM**

#include<iostream>

using namespace std;

struct Node

{

int processor, Execution\_time;

Node \*Next\_Node;

};

class Round\_Robin

{

public:

Node \*front, \*rear; // Initillizing Elements

int Quantum\_Time;

int Total\_Time;

int Average\_Time;

Round\_Robin()

{

Quantum\_Time = 2, Total\_Time = 0, Average\_Time = 0;

front = NULL, rear = NULL;

}

void enqueue(int proc, int exec) //ENQUEUE FUNCTION

{

Node \*temp = new Node;

if (front == NULL && rear == NULL)

{

temp->processor = proc;

temp->Execution\_time = exec;

rear = temp;

front = temp;

temp->Next\_Node = front;

}

else

{

temp->processor = proc;

temp->Execution\_time = exec;

rear->Next\_Node = temp;

rear = temp;

temp->Next\_Node = front;

}

}

void processing()

{

Node \*current = front, \*temp = new Node, \*temp1 = new Node;

while (current != NULL)

{

current = front;

cout << endl << "Processor : " << front->processor << "\tExecution Time : " << front->Execution\_time << "\t Quantum Time :" << Quantum\_Time << endl;

if (current->Execution\_time >= 2)

{

current->Execution\_time = current->Execution\_time - Quantum\_Time;

temp->Execution\_time = current->Execution\_time;

temp->processor = current->processor;

if (front != rear)

{

front = front->Next\_Node;

rear->Next\_Node = temp;

temp->Next\_Node = front; // if time is greater than 2 so we'll minus 2 from it and

rear = temp; // take that node to last and delete it from first

temp1 = current;

free(temp1);

Total\_Time = Total\_Time + 2;

}

else

{

front = temp;

rear = temp;

temp->Next\_Node = front;

temp1 = current;

free(temp1);

Total\_Time = Total\_Time + 2;

}

}

else if (current->Execution\_time == 1)

{

current->Execution\_time = current->Execution\_time - 1;

temp->Execution\_time = current->Execution\_time;

temp->processor = current->processor;

if (front != rear)

{

front = front->Next\_Node;

rear->Next\_Node = temp;

temp->Next\_Node = front; //if execution value comes out to be 1 then we'll minus 1 from

rear = temp; // it and follow the same step

temp1 = current;

free(temp1);

Total\_Time = Total\_Time + 1;

}

else

{

free(current);

front = NULL, rear = NULL; //if front is equal to rear than it means it is the last one

Total\_Time = Total\_Time + 1; // hence we'll break the loop

break;

}

}

else if (current->Execution\_time == 0)

{

if (front == rear)

{

front = NULL;

rear = NULL;

free(current); // if after minus value just remains 0 then break if front == rear

break;

}

if (front != rear)

{

front = front->Next\_Node;

temp1 = current;

free(temp1);

}

}

current = current->Next\_Node;

}

cout << endl << "Total Time taken = " << Total\_Time;

}

};

int main()

{

Round\_Robin queue;

int Sentinal = 0, Processors = 0, Time\_Duration = 0;

while (Sentinal != 1)

{

system("cls");

cout << "Enter The Of Processors : ";

cin >> Processors;

cout << "Enter The Time Duration For Given Processors : ";

cin >> Time\_Duration;

queue.enqueue(Processors, Time\_Duration);

cout << endl << "Enter 1 to Check Time Duration of Provided Processors " << endl;

cout << " OR " << endl;

cout << "Enter Any Number to Repeat the Same Process : ";

cin >> Sentinal;

}

system("cls");

queue.processing();

cout << endl << endl;

system("pause");

}

A screenshot of a computer screen

Description automatically generated