# TASK 1:

#include<iostream>

#include<string>

using namespace std;

struct node

{

string name;

node\* next;

node()

{

next = NULL;

}

node(string x) :node()

{

name = x;

}

};

class Sanet

{

node\* head;

public:

Sanet()

{

head = NULL;

}

void Insert(string x)

{

node\* newnode = new node(x);

if (head == NULL)

{

head = newnode;

newnode->next = head;

return;

}

node\* curr = head;

while (curr->next != head)

{

curr = curr->next;

}

curr->next = newnode;

newnode->next = head;

}

bool pop(node\* curr)

{

if (curr->next == head && curr == head)

{

return 0;

}

node\* temp = curr->next;

curr->next = temp->next;

delete temp;

return 1;

}

void win()

{

char x;

bool che;

node\* curr = head;

while (true)

{

cout << "Do you want to remove next senate.... [Y|N] : ";

cin >> x;

if (x == 'y' || x == 'Y')

{

che = pop(curr);

if (che == 0)

{

cout << curr->name << " Is the winner " << endl;

return;

}

}

curr = curr->next;

}

}

};

int main()

{

Sanet obj;

string name;

for (int i = 0; i < 6; i++)

{

cout << "Enter name : ";

cin >> name;

obj.Insert(name);

}

obj.win();

cout << endl;

system("pause");

}

# OUTPUT:

# 

# TASK 2:

#include<iostream>

using namespace std;

template<class T>

class queue

{

private:

T name[10];

int front, rare;

public:

bool enqueue(T data)

{

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

{

front = rare = 0;

name[rare] = data;

return true;

}

else

{

if (isfull())

{

name[rare] = data;

return true;

}

}

return true;

}

bool isfull()

{

if (front == 0 && rare == 9)

return false;

else

{

rare++;

if (rare == 10)

rare = 0;

if (rare == front)

return false;

return true;

}

}

bool isEmpty()

{

if (rare == -1)

return false;

return true;

}

T dequeu(T data)

{

T temp = name[front];

name[front] = data;

front++;

return temp;

}

void functionality(queue <string>arr[], int size)

{

arr[size - 1].dequeu("0");

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

{

arr[size - i].enqueue(arr[size - (i + 1)].dequeu(" 0"));

}

}

void display(queue <string>arr[], int size)

{

for (int i = 0; i < size; i++)

cout << " " << i + 1 << " ";

cout << endl;

int temp = arr[1].front;

do

{

for (int i = 0; i < size; i++)

{

cout << arr[i].name[temp] << " ";

}

cout << endl;

temp++;

if (temp == 10)

temp = 0;

} while (temp != arr[1].front);

}

queue()

{

front = rare = -1;

}

};

int main()

{

int num;

cout << "Enter number of queues:";

cin >> num;

queue<string>\* obj, temp;

obj = new queue<string>[num];

for (int i = 0; i < num; i++)

{

int c = 48;

for (int j = 0; j < 10; j++, c++)

{

string str = "person";

obj[i].enqueue(str + char(c));

}

}

cout << "You have following queues and person in that queues" << endl;

temp.display(obj, num);

char choice;

do

{

cout << "If you want to process person press y/Y else press any key:";

cin >> choice;

if (choice == 'y' || choice == 'Y')

temp.functionality(obj, num);

else

break;

} while (1);

temp.display(obj, num);

cout << "0 means this is person is shifted:";

system("pause");

return 0;

}

# OUTPUT:

# 

# TASK 3:

#include <iostream>

#include <cmath>

using namespace std;

int binaryToDecimal(int binaryNumber)

{

int decimal = 0, i = 0, remainder;

while (binaryNumber != 0)

{

remainder = binaryNumber % 10;

binaryNumber /= 10;

decimal += remainder \* pow(2, i);

++i;

}

return decimal;

}

int main()

{

int binary;

cout << "Enter a binary number: ";

cin >> binary;

cout << binary << " in binary = " << binaryToDecimal(binary) << " in decimal" << endl;

return 0;

}

# OUTPUT:

