DSA GROUP PROJECT

|  |  |
| --- | --- |
| **Student name** | **Student number** |
| Owen Luyando Chilwalo | 222038578 |
| Taloshili Nghiwewelekwa | 221057730 |
| Asante-Sana Maruwasa | 222073470 |
| Omukwathi Mbandeka | 221028552 |
| Paul Nkondo | 222107502 |
| Rauana Murangi | 217149022 |
| Emily Manjara | 222032227 |
| Martin Nakatana | 222007265 |
| Ricky Fangda | 221140085 |
| Maria Shikwambi | 222064374 |

SECTION A

Pseudocode

Start

FUNCTION playFwd()

head=1 //1st node/song;

WHILE(head!=null)

PRINT head.data

head = head.nextaddress

ENDWHILE

END FUNCTION

FUNCTION playRepeat()

stop = false

head=1

WHILE(head!=null)

PRINT head.data

Prompt to stop

input stop

if(stop == TRUE) THEN

BREAK

end if

head = head.nextaddress

ENDWHILE

END FUNCTION

FUNCTION playBack()

head=last  //last node/song in the list

WHILE(head!=null)

PRINT head.data

head = head.prevaddress

ENDWHILE

END FUNCTION

FUNCTION addSong()

IF (head = null) THEN

head = 0

INPUT song

songNum = head

head.data = song

head.nextaddress = null

prev = head

ELSEIF (head==prev) THEN

head = head.nextaddress

INPUT song

songNum = head

head.data = song

head.nextaddress = null

prev = head

ENDIF

END FUNCTION

FUNCTION search()

INPUT search

head=1

while(head!=null)

IF (head==search)THEN

Play head.data  //Print or play song?

ELSE

Print "Song not found"

ENDIF

ENDWHILE

END FUNCTION

FUNCTION RemoveTracks()

Set p = Value2

if(head==value2 and p->pointer !Null)

delete current

else if(p->pointer !=Null and p->previousAddressPart)

p->previousAddressPart = p->pointer

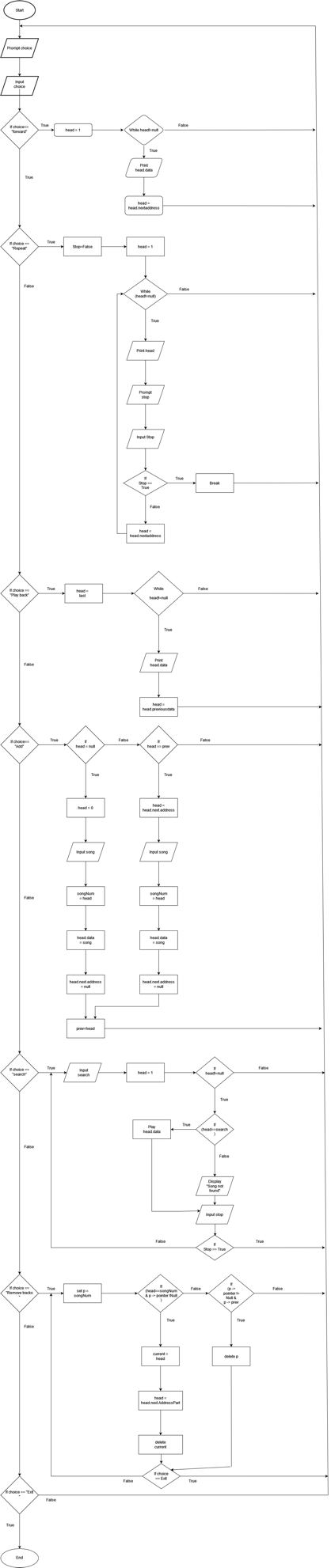
delete p

ENDIF

END FUNCTION

End

Flowchart



Section B

import java.util.\*;

public class Main {

static Node begin;

static class Node {

String data;

Node next;

Node previous;

}

static Node addSong(String song)

{

if (begin == null) {

Node new\_node = new Node();

new\_node.data = song;

new\_node.next = new\_node.previous = new\_node;

begin = new\_node;

return begin;

}

Node last = (begin).previous;

Node new\_node = new Node();

new\_node.data = song;

new\_node.next = begin;

(begin).previous = new\_node;

new\_node.previous = last;

last.next = new\_node;

return begin;

}

static Node removeSong(Node begin, String song)

{

if (begin == null)

return null;

Node current = begin, previous\_node = null;

while (!current.data.equals( song)) {

if (current.next == begin) {

System.out.print("\nPlaylist does not have the song with name: " + song);

return begin;

}

previous\_node = current;

current = current.next;

}

if (current.next == begin && previous\_node == null) {

(begin) = null;

return begin;

}

if (current == begin) {

previous\_node = (begin).previous;

begin = (begin).next;

previous\_node.next = begin;

(begin).previous = previous\_node;

}

// check if it is the last node

else if (current.next == begin) {

previous\_node.next = begin;

(begin).previous = previous\_node;

}

else {

Node temp = current.next;

previous\_node.next = temp;

temp.previous = previous\_node;

}

return begin;

}

static String searchSong(Node begin, String song)

{

Node temp = begin;

int count = 0, found = 0;

if(temp == null)

return song;

else

{

while(!temp.next.equals(begin))

{

count++;

if(temp.data.equals( song))

{

found = 1;

count--;

break;

}

temp = temp.next;

}

if(temp.data.equals(song))

{

count++;

found = 1;

}

if(found == 1)

System.out.println("\n"+song +" found at the position "+

count);

else

System.out.println("\n"+song +" was not found");

}

return song;

}

static void play(Node begin)

{

Node temp = begin;

System.out.print(

"Playing playlist");

while (!temp.next.equals(begin)) {

System.out.println("playing " + temp.data);

temp = temp.next;

}

System.out.println("playing " + temp.data);

System.out.println(

"Playing playlist in reverse direction \n");

Node last = begin.previous;

temp = last;

while (temp.previous != last) {

System.out.println("playing" + temp.data);

temp = temp.previous;

}

System.out.println("playing" + temp.data);

}

static void repeat(Node begin, int count){

Node temp = begin;

//boolean running = true;

System.out.println(

"\n!Playing playlist! \n");

for(int i = 0; i < count; i++){

System.out.println("playing " + temp.data);

temp = temp.next;}

}

public static void main(String[] args)

{

Scanner input = new Scanner(System.in);

Node begin = null;

int choice ;

String song\_name;

boolean running = true;

while(running){

System.out.println("Enter Choice \n1. Add Song to playlist \n2. Remove Song from playlist \n3. Search Song in playlist \n4. play playlist \n5. repeat playlist \n6. Quit");

choice = input.nextInt();

if (choice == 1){

System.out.println("Enter song name to be added to playlist: ");

song\_name = input.next();

begin = addSong(song\_name);

System.out.println("Successfully Added ");

}

else if (choice == 2){

System.out.println("enter song name to be removed from playlist:");

song\_name = input.next();

begin = removeSong(begin,song\_name);

}

else if (choice == 3){

System.out.println("Enter the name of the song to be searched in the playlist");

song\_name = input.next();

searchSong(begin,song\_name);

}

else if (choice == 4){

play(begin);

}

else if (choice == 5){

System.out.println("Enter the amount of times that the playlist must repeat itself");

int count = input.nextInt();

repeat(begin, count);

}

else if (choice == 6) {

running = false;

}

}

}

}