#include<iostream>

using namespace std;

struct node

{

int data;

node\* next;

node\* prev;

};

node\* head = NULL;

node\* tail = NULL;

void insert(int data)

{

node\* temp = new node;

temp->data = data;

temp->next = NULL;

temp->prev = NULL;

if (head == NULL)

{

head = temp;

tail = temp;

}

else

{

tail->next = temp;

temp->prev = tail;

tail = temp;

}

}

void bubble\_Sort()

{

node\* tempHead, \* tempEnd = NULL;

bool swaping = true;

while (swaping == true)

{

swaping = false;

tempHead = head;

while (tempHead -> next != tempEnd)

{

if (tempHead -> data > tempHead -> next -> data)

{

int tempNum = tempHead -> data;

tempHead -> data = tempHead -> next -> data;

tempHead -> next -> data = tempNum;

swaping = true;

}

tempHead = tempHead -> next;

}

tempEnd = tempHead;

}

tempEnd = tempHead = NULL; delete tempEnd;

delete tempHead;

}

void deletes()

{

node\* temp2 = new node;

temp2 = head;

while (tail->next != NULL)

{

tail = tail->next;

}

temp2 = tail->prev;

temp2->next = NULL;

delete tail;

}

void search(int x)

{

node\* temp = head;

while (temp->next != NULL)

{

if (temp->data == x)

{

cout << "\nNumber has been found"<<endl;

break;

}

else

{

cout << "\nNumber has not been found"<<endl;

break;

}

temp = temp->next;

}

}

void insert(int x, int pos)

{

node\* temp = new node;

temp->data = x;

temp->next = NULL;

temp->prev = NULL;

if (pos == 1)

{

temp->next = head;

head->prev = temp;

temp->prev = NULL;

head = temp;

}

else

{

node\* temp2 = new node;

temp2 = head;

for (int i = 1; i < pos - 1; i++)

{

temp2 = temp2->next;

}

temp->next = temp2->next;

temp2->next = temp;

temp->prev = temp2;

}

}

void display()//function to display the linked list

{

node\* temp = head;

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int main()

{

int data,x;

int position, data1;

int num;

cout << "how many nodes you want to add :";

cin >> x;

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

{

cin >> data;

insert(data);

}

cout << "Element after sorting are :";

bubble\_Sort();

display();

cout << "\nElement after deleting are :";

deletes();

display();

cout << "\nEnter number you want to search :";

cin >> num;

search(num);

cout << "\nEnter position on which you want to add the new node";

cin >> position;

cout << "\nEnter data for new node";

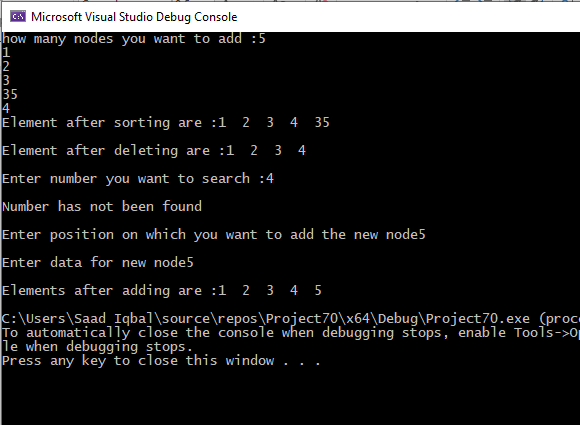
cin >> data1;

insert(data1, position);

cout << "\nElements after adding are :";

display();

}



TASK 2

#include<iostream>

using namespace std;

struct node

{

int data;

node\* next;

};

node\* head = NULL;

node\* tail = NULL;

void insert(int data)

{

node\* temp = new node;

temp->data = data;

temp->next = NULL;

if (head == NULL)

{

head = temp;

tail = temp;

}

else

{

tail->next = temp;

tail = tail->next;

}

}

void swap(struct node\* list, int n)

{

struct node\* node1=NULL, \* node2=NULL, \* prev1, \* prev2, \* temp;

int i;

int maxPos;

int pos1, pos2;

a: cout << "Enter position 1 to replace the nodes :";

cin >> pos1;

cout << "Enter position 2 to replace the nodes :";

cin >> pos2;

if (pos1 > pos2)

{

maxPos = pos1;

}

else

{

maxPos = pos2;

}

const int totalNodes = n;

if ((pos1 <= 0 || pos1 > totalNodes) || (pos2 <= 0 || pos2 > totalNodes))

{

cout << "Enter position again" << endl;

goto a;

}

i = 1;

temp = list;

prev1 = NULL;

prev2 = NULL;

while (temp != NULL)

{

if (i == pos1 - 1)

{

prev1 = temp;

}

if (i == pos1)

{

node1 = temp;

}

if (i == pos2 - 1)

{

prev2 = temp;

}

if (i == pos2)

{

node2 = temp;

}

temp = temp->next;

i++;

}

if (node1 != NULL && node2 != NULL)

{

if (prev1 != NULL)

{

prev1->next = node2;

}

if (prev2 != NULL)

{

prev2->next = node1;

}

temp = node1->next;

node1->next = node2->next;

node2->next = temp;

if (prev1 == NULL)

{

head = node2;

}

else if (prev2 == NULL)

{

head = node1;

}

}

}

void display()

{

node\* temp = head;

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int main()

{

int num, data, pos1, pos2;

node\* list = NULL;

cout << "Enter how many nodes you want to add :";

cin >> num;

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

{

cout << "Enter data of node " << i + 1 << " :";

cin >> data;

insert(data);

}

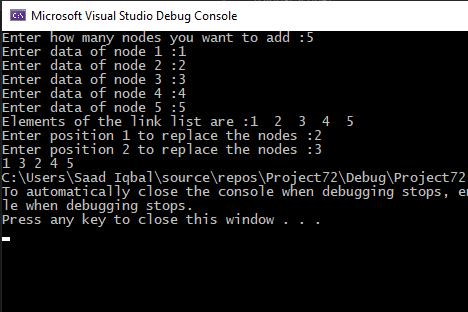
cout << "Elements of the link list are :";

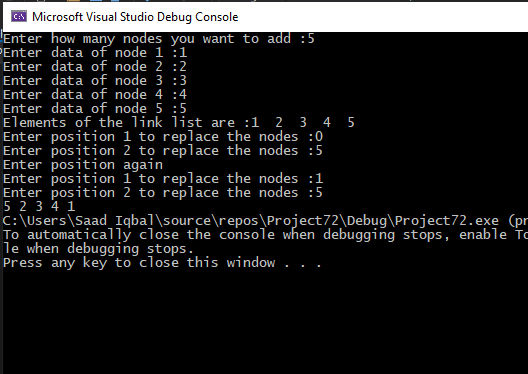
display();

swap(list, num);

display();

}





TASK 3:

#include<iostream>

using namespace std;

struct node

{

int data;

node\* next;

};

node\* head = NULL;

node\* tail = NULL;

void insert(int data)

{

node\* temp = new node;

temp->data = data;

temp->next = NULL;

if (head == NULL)

{

head = temp;

tail = temp;

}

else

{

tail->next = temp;

tail = tail->next;

}

}

void reverse() //function to reverse linked list

{

node\* temp = head;

node\* tail = NULL;

node\* next = NULL;

while (temp != NULL)

{

next = temp->next;

temp->next = tail;

tail = temp;

temp = next;

}

head = tail;

}

void display()

{

node\* temp = head;

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int main()

{

int num, data, pos1, pos2;

node\* list = NULL;

cout << "Enter how many nodes you want to add :";

cin >> num;

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

{

cout << "Enter data of node " << i + 1 << " :";

cin >> data;

insert(data);

}

cout << "Elements of the link list are :";

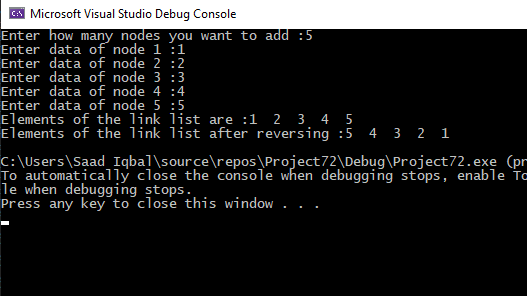
display();

reverse();

cout << "Elements of the link list after reversing :";

display();

}



TASK 4:

#include<iostream>

using namespace std;

struct node

{

int data;

node\* next;

node\* prev;

};

node\* head = NULL;

node\* tail = NULL;

void insert(int data)

{

node\* temp = new node;

temp->data = data;

temp->next = NULL;

temp->prev = NULL;

if (head == NULL)

{

head = temp;

tail = temp;

}

else

{

tail->next = temp;

temp->prev = tail;

tail = temp;

}

}

void input(int n,int num)

{

node\* temp=head;

node\* ttail = tail;

int c = 0;

while (ttail != NULL)

{

if (temp->data + ttail->data == n)

{

cout << temp->data << "+" << ttail->data << ",";

temp = temp->next;

}

else

{

c++;

}

if (c == num)

{

cout << "PAIR NOT FOUND";

}

ttail = ttail->prev;

}

}

void display()

{

node\* temp = head;

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

int main()

{

int num, data, pos1, pos2;

node\* list = NULL;

int n;

cout << "Enter how many nodes you want to add :";

cin >> num;

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

{

cout << "Enter data of node " << i + 1 << " :";

cin >> data;

insert(data);

}

cout << "Elements of the link list are :";

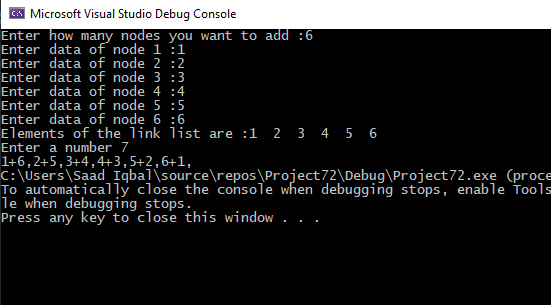
display();

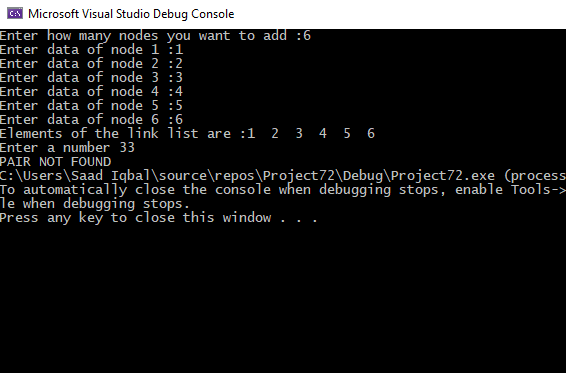
cout << "Enter a number ";

cin >> n;

input(n,num);

}





**STACK**

TASK 1:

#include<iostream>

#include<stack>

using namespace std;

struct node

{

int data;

node\* next;

};

class ArrayStack

{

public:

int top;

int a[100];

ArrayStack()

{

top = -1;

}

bool push(int x)

{

if (top >= (100 - 1))

{

cout << "Stack Overflow";

return false;

}

else

{

top++;

a[top] = x;

cout << x << " pushed into stack\n";

return true;

}

}

int pop()

{

if (top < 0)

{

cout << "Stack Underflow";

return 0;

}

else

{

int x = a[top--];

return x;

}

}

int peek()

{

if (top < 0)

{

cout << "Stack is Empty";

return 0;

}

else

{

int x = a[top];

return x;

}

}

bool isEmpty()

{

return (top < 0);

}

void display()

{

if (top >= 0)

{

cout << "\nStack elements are:";

for (int i = top; i >= 0; i--)

{

cout << a[i] << " ";

}

cout << endl;

}

else

cout << "Stack is empty";

}

void frequent()

{

int count = 0,n=0;

for (int i = top; i >= 0; i--)

{

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

{

if (a[i] == a[j])

{

count++;

n = a[i];

}

}

}

cout << endl;

cout << n<<" is the most frequent number";

}

};

int main()

{

ArrayStack lls;

int num,data;

cout << "How many numbers you want to push in a stack :";

cin >> num;

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

{

cout << "Pushin to the stack " << i + 1<<" :";

cin >> data;

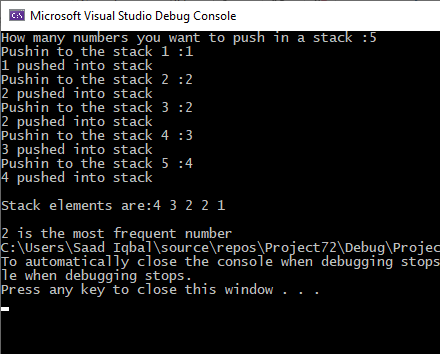
lls.push(data);

}

lls.display();

lls.frequent();

}



TASK 2:

#include<iostream>

using namespace std;

struct node

{

int data;

node\* NEXT;

}\*heading;

class queue

{

int size;

node\* temp;

node\* tail;

public:

queue()

{

heading = NULL;

tail = NULL;

size = 0;

}

~queue() {}

bool isempty()

{

if (heading == NULL)

{

return 1;

}

else

{

return 0;

}

}

void enqueue(int data)

{

node\* newnode = new node;

newnode->data = data;

newnode->NEXT = heading;

heading = newnode;

size--;

}

void dequeue()

{

if (isempty())

{

cout << "STACK is Empty" << endl;

}

node\* temp = heading;

heading = (heading)->NEXT;

free(temp);

}

void display()

{

if (heading == NULL)

{

cout << "STACK is empty \n";

}

else

{

node\* ptr = heading;

cout << "STACK elements are: ";

while (ptr != NULL)

{

cout << ptr->data << " ";

ptr = ptr->NEXT;

}

cout << endl;

}

}

};

int main()

{

int n;

queue access;

cout << "WELCOME TO Stack WITH queue, WHAT YOU WANT TO DO ?" << endl;

a: cout << " 1.Check if STACK is empty.\n 2. Enter data in STACK \n 3. Dequeue data \n 4. Display the data \n 5. to exit \n";

cin >> n;

if (n == 1)

{

if (access.isempty()) {

cout << "STACK is empty" << endl;

}

else {

cout << "STACK is not empty" << endl;

access.display();

}

goto a;

}

if (n == 2)

{

int data, n;

cout << "Enter the number of element you add in STACK" << endl;

cin >> n;

cout << "ENTER THE DATA YOU WANT TO ADD IN STACK \n";

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

{

cin >> data;

access.enqueue(data);

}

goto a;

}

if (n == 3)

{

int n;

cout << "Enter the number of element you want to pop" << endl;

cin >> n;

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

{

access.dequeue();

}

goto a;

}

if (n == 4)

{

access.display();

goto a;

}

if (n == 5)

{

goto z;

}

else

{

cout << "WRONG INPUT \n";

goto a;

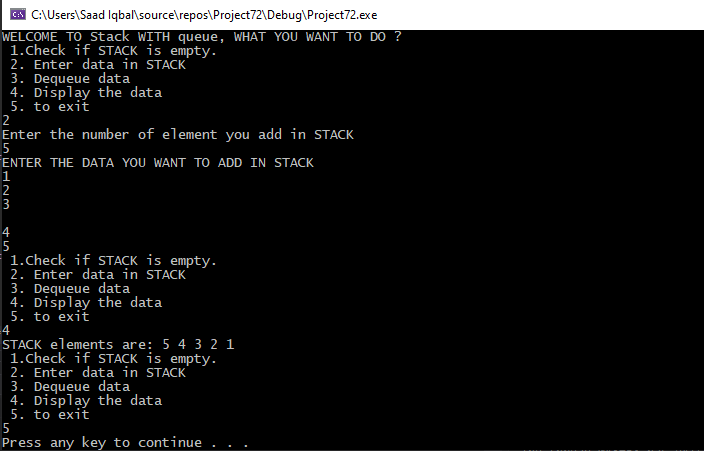
}

z:

system("pause");

return 0;

}



TASK 3:

#include<iostream>

#include<string>

using namespace std;

struct node

{

string data;

node\* next;

};

class ArrayStack

{

public:

int top;

string a[100];

ArrayStack()

{

top = -1;

}

bool push(string x)

{

if (top >= (100 - 1))

{

cout << "Stack Overflow";

return false;

}

else

{

top++;

a[top] = x;

return true;

}

}

string pop()

{

if (top < 0)

{

cout << "Stack Underflow";

return " ";

}

else

{

string x = a[top--];

return x;

}

}

string peek()

{

if (top < 0)

{

cout << "Stack is Empty";

return " ";

}

else

{

string x = a[top];

return x;

}

}

bool isEmpty()

{

return (top < 0);

}

void display()

{

if (top >= 0)

{

cout << "\nStack elements are:";

for (int i = top; i >= 0; i--)

{

cout << a[i] << " ";

}

cout << endl;

}

else

cout << "Stack is empty";

}

void Repeated()

{

int count = 0;

string n = " ";

for (int i = top; i >= 0; i--)

{

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

{

if (a[i] == a[j])

{

a[i].clear();

}

}

}

cout << "Stack after deleteing repeating elements :";

for (int j = 0; j < top; j++)

{

cout << a[j] << " ";

}

}

};

int main()

{

ArrayStack lls;

int num;

string data;

cout << "How many numbers you want to push in a stack :";

cin >> num;

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

{

cout << "Pushin to the stack " << i + 1 << " :";

cin >> data;

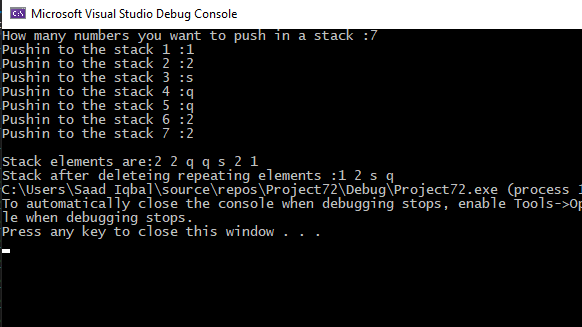
lls.push(data);

}

lls.display();

lls.Repeated();

}



Queue

TASK 1:

#include<iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

}\*heading;;

class queue

{

int size;

Node\* temp;

Node\* tail;

public:

queue()

{

heading = NULL;

tail = NULL;

size = 0;

}

~queue()

{}

void enqueue(int data)

{

temp = new Node;

temp->data = data;

temp->next = NULL;

if (heading == NULL)

{

tail = temp;

heading = temp;

}

else

{

tail->next = temp;

tail = temp;

}

size++;

}

void display()

{

if (heading == NULL)

{

cout << "Queue is empty \n";

}

else

{

Node\* ptr = heading;

cout << "Queue elements are: front->";

while (ptr != NULL)

{

cout << ptr->data << "->";

ptr = ptr->next;

}

cout << "rear";

cout << endl;

}

}

Node\* reverse(Node\* node)

{

if (node == NULL)

return NULL;

if (node->next == NULL) {

heading = node;

return node;

}

Node\* node1 = reverse(node->next);

node1->next = node;

node->next = NULL;

return node;

}

};

int main()

{

queue g;

int op;

a: cout << "Press 1 to enqueue values :" << endl;

cout << "Press 2 to display values :" << endl;

cout << "Press 3 to reverse values :" << endl;

cout << "Press 4 to exit the program:" << endl;

cin >> op;

while (1)

{

if (op == 1)

{

g.enqueue(4);

g.enqueue(6);

g.enqueue(2);

g.enqueue(5);

g.enqueue(3);

goto a;

}

if (op == 2)

{

g.display();

goto a;

}

if (op == 3)

{

int n;

cout << "Enter number you want to reverse :";

cin >> n;

g.reverse(heading);

cout << "After reversing :" << endl;

g.display();

goto a;

}

if (op == 4)

{

break;

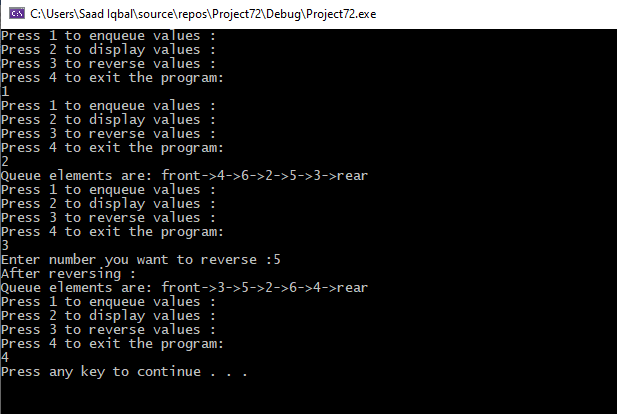
}

}

system("pause");

return 0;

}



TASK 2:

#include<iostream>

using namespace std;

struct node

{

int data;

node\* NEXT;

}\*heading;

class queue

{

int size;

node\* temp;

node\* tail;

public:

queue()

{

heading = NULL;

tail = NULL;

size = 0;

}

~queue() {}

bool isempty()

{

if (heading == NULL)

{

return 1;

}

else

{

return 0;

}

}

void enqueue(int data)

{

node\* newnode = new node;

newnode->data = data;

newnode->NEXT = heading;

heading = newnode;

size++;

}

void dequeue()

{

if (isempty())

{

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

}

node\* temp = heading;

heading = (heading)->NEXT;

free(temp);

}

void display()

{

if (heading == NULL)

{

cout << "STACK is empty \n";

}

else

{

node\* ptr = heading;

cout << "STACK elements are: ";

while (ptr != NULL)

{

cout << ptr->data << " ";

ptr = ptr->NEXT;

}

cout << endl;

}

}

};

int main()

{

int n;

queue access;

cout << "WELCOME TO QUEUE WITH Stack, WHAT YOU WANT TO DO ?" << endl;

a: cout << " 1.Check if queue is empty.\n 2. Enter data in queue \n 3. Dequeue data \n 4. Display the data \n 5. to exit \n";

cin >> n;

if (n == 1)

{

if (access.isempty()) {

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

}

else {

cout << "Queue is not empty" << endl;

access.display();

}

goto a;

}

if (n == 2)

{

int data, n;

cout << "Enter the number of element you add in queue" << endl;

cin >> n;

cout << "ENTER THE DATA YOU WANT TO ADD IN QUEUE \n";

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

{

cin >> data;

access.enqueue(data);

}

goto a;

}

if (n == 3)

{

int n;

cout << "Enter the number of element you want to dequeue" << endl;

cin >> n;

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

{

access.dequeue();

}

goto a;

}

if (n == 4)

{

access.display();

goto a;

}

if (n == 5)

{

goto z;

}

else

{

cout << "WRONG INPUT \n";

goto a;

}

z:

system("pause");

return 0;

}

