**ALI HASSAN 03-135211-005**

**ASSIGNMENT 4**

**TASK 1:**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

};

void create(Node\*\*head, int val) {

Node\* new\_node = new Node();

new\_node->data = val;

new\_node->next = \*head;

\*head = new\_node;

}

void display\_list(Node \*head) {

while (head->next != NULL) {

cout << head->data << " -> ";

head = head->next;

}

}

void middleNode(Node\* head) {

Node\* fast\_ptr;

Node\* slow\_ptr;

fast\_ptr = head;

slow\_ptr = head;

if (head != NULL) {

while (fast\_ptr->next != NULL && slow\_ptr->next != NULL) {

fast\_ptr = fast\_ptr->next->next;

slow\_ptr = slow\_ptr->next;

}

cout <<endl << "Middle element is : " << slow\_ptr->data;

}

}

int main() {

Node\* head = new Node();

const int size = 10;

int arr[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

for (int i = size - 1; i >= 0; i--) {

create(&head, arr[i]);

}

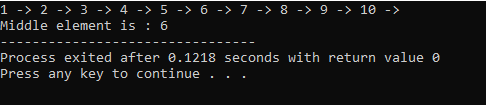
display\_list(head);

middleNode(head);

return 0;

}

**OUTPUT:**

****

**TASK 2:**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

};

void create(Node\*\*head, int val) {

Node\* new\_node = new Node();

new\_node->data = val;

new\_node->next = \*head;

\*head = new\_node;

}

void display\_list(Node \*head) {

while (head->next != NULL) {

cout << head->data << " -> ";

head = head->next;

}

}

Node\* circularlist(Node\* head) {

Node\* new\_head = head;

while (head->next != NULL) {

head = head->next;

}

head->next = new\_head;

return new\_head;

}

int main() {

Node\* head = NULL;

const int size = 3;

int arr[size] = { 1, 2, 3 };

for (int i = size - 1; i >= 0; i--) {

create(&head, arr[i]);

}

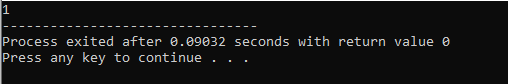
head = circularlist(head);

cout << head->next->next->next->data;

return 0;

}

**Output:**

****

**Task 3:**

#include <iostream>

using namespace std;

#define stack\_size 5

class Stack {

public:

int top = -1;

int arr[stack\_size];

//Stack();

//Stack(int size);

void push(int);

int pop();

bool isEmpty();

bool isFull();

int getTop();

};

void Stack :: push(int val) {

if (top >= stack\_size) {

cout << "Stack is full already!" << endl;

} else {

arr[++top] = val;

}

}

int Stack::pop() {

int el = 0;

if (top < 0) {

cout << "no elements to remove!" << endl;

}

else {

el = arr[top--];

}

return el;

}

bool Stack::isEmpty() {

if (top < 0)

return true;

else

return false;

}

bool Stack::isFull() {

if (top == 4)

return true;

else

return false;

}

int Stack::getTop(){

return arr[top];

}

int main() {

Stack st;

st.push(5);

st.push(3);

cout << "removed value from stack: " << st.pop() << endl;

if (st.isEmpty())

cout << "Stack is empty!" << endl;

if (st.isFull())

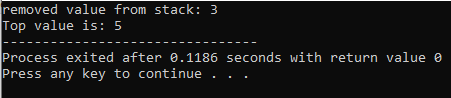
cout << "Stack is full!" << endl;

cout << "Top value is: " << st.getTop();

return 0;

}

**Output:**

****

**Task 4:**

#include <iostream>

using namespace std;

#define stack\_size 5

class Stack {

public:

int top = -1;

int arr[stack\_size];

//Stack();

//Stack(int size);

void push(int);

int pop();

bool isEmpty();

bool isFull();

int getTop();

};

void Stack::push(int val) {

if (top >= stack\_size) {

cout << "Stack is full already!" << endl;

}

else {

arr[++top] = val;

}

}

int Stack::pop() {

int el = 0;

if (top < 0) {

cout << "no elements to remove!" << endl;

}

else {

el = arr[top--];

}

return el;

}

bool Stack::isEmpty() {

if (top < 0)

return true;

else

return false;

}

bool Stack::isFull() {

if (top == 4)

return true;

else

return false;

}

int Stack::getTop() {

return arr[top];

}

int main() {

Stack st, st2, st3;

st.push(5);

st.push(3);

st2.push(st.pop());

st2.push(st.getTop());

st3.push(st2.pop());

st3.push(st2.getTop());

cout << "Copied to second stack: ";

cout << st3.pop();

cout << st3.getTop();

return 0;

}

**Output**:

