**ALI HASSAN 03-135211-005**

**ASSIGNMENT 6**

**TASK 1:**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

};

Node\* top = NULL;

void push(int val) {

Node\* temp = new Node();

temp->data = val;

temp->next = top;

top = temp;

}

void display() {

Node\* temp = top;

if (temp == NULL) {

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

}

else {

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

}

}

int getTop() {

if (top == NULL)

cout << "Stack is empty!";

else

return top->data;

return 0;

}

int pop() {

if (top == NULL) {

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

}

else {

cout << "Value deleted: " << top->data << endl;

top = top->next;

}

return -1;

}

bool isEmpty() {

if (top == NULL)

return true;

return false;

}

int main() {

int i = 0;

while (i <= 7) {

push(++i);

}

display();

cout << endl;

pop();

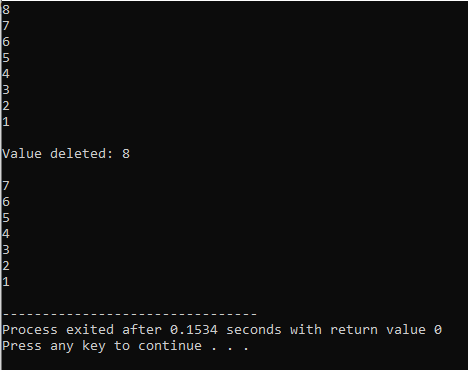
cout << endl;

display();

return 0;

}

**Output:**

****

**Task 2:**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

};

Node\* front = NULL;

Node\* back = NULL;

void enQueue(int val) {

Node\* temp = new Node();

temp->data = val;

temp->next = NULL;

if (back == NULL) {

front = temp;

back = temp;

}

back->next = temp;

back = temp;

}

void display() {

Node\* temp = front;

while (temp->next != NULL) {

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

temp = temp->next;

}

cout << temp->data << endl;

}

void deQueue() {

Node\* temp\_front = front;

front = front->next;

(temp\_front);

}

void front\_val() {

cout << front->data;

}

int main() {

cout << "qeueue: " << endl;

enQueue(1);

enQueue(2);

enQueue(3);

enQueue(4);

display();

cout << endl;

cout << "dequeued: " << endl;

deQueue();

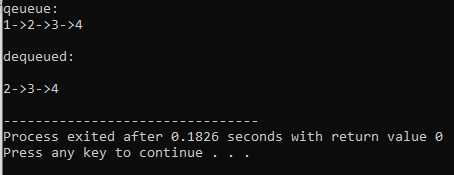
cout << endl;

display();

return 0;

}

**Output:**

****

**Task 3:**

#include <iostream>

using namespace std;

// stack implementation using dynamic linked list

class Node {

public:

int data;

Node\* next;

};

Node\* top = NULL;

int total = 0;

void push(int val) {

Node\* temp = new Node();

temp->data = val;

temp->next = top;

top = temp;

total++;

}

void display() {

Node\* temp = top;

if (temp == NULL) {

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

}

else {

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next;

}

}

}

int getTop() {

if (top == NULL)

cout << "Stack is empty!";

else

return top->data;

return 0;

}

int pop() {

if (top == NULL) {

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

}

else {

cout << "Value deleted: " << top->data << endl;

top = top->next;

total--;

}

return -1;

}

bool isEmpty() {

if (top == NULL)

return true;

return false;

}

void findCenter() {

Node\* fast\_ptr = top;

Node\* slow\_ptr = top;

if (top != NULL) {

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

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

slow\_ptr = slow\_ptr->next;

}

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

}

}

int main() {

int i = 0;

while (i < 7) {

push(++i);

}

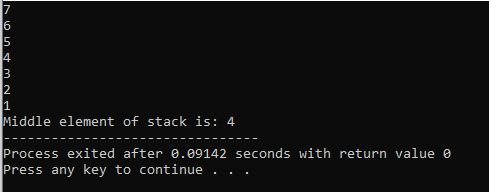
display();

findCenter();

return 0;

}

**Output:**

****

**Task 4:**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

};

Node\* front = NULL;

Node\* back = NULL;

int total = 0;

int\* arr = new int[total + 1];

void enQueue(int val) {

Node\* temp = new Node();

temp->data = val;

temp->next = NULL;

if (back == NULL) {

front = temp;

back = temp;

}

back->next = temp;

back = temp;

total++;

}

void display() {

Node\* temp = front;

while (temp->next != NULL) {

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

temp = temp->next;

}

cout << temp->data << endl;

}

void deQueue() {

Node\* temp\_front = front;

front = front->next;

free(temp\_front);

total--;

}

void front\_val() {

cout << front->data;

}

int totalVal() {

return total;

}

void storeToArray() {

Node\* temp = front;

int i = -1;

//cout << temp->next->next->next->data;

while (temp->next != NULL) {

arr[++i] = temp->data;

temp = temp->next;

}

arr[++i] = temp->data;

for (int j = 0; j < total; j++) {

cout << arr[j] << " ";

}

}

int main() {

cout << "Qeueue: " << endl;

enQueue(1);

enQueue(2);

enQueue(3);

enQueue(4);

display();

cout << endl;

cout << "Total Values: " << totalVal();

cout << endl;

cout << "Array values: " << endl;

storeToArray();

return 0;

}

**Output:**

