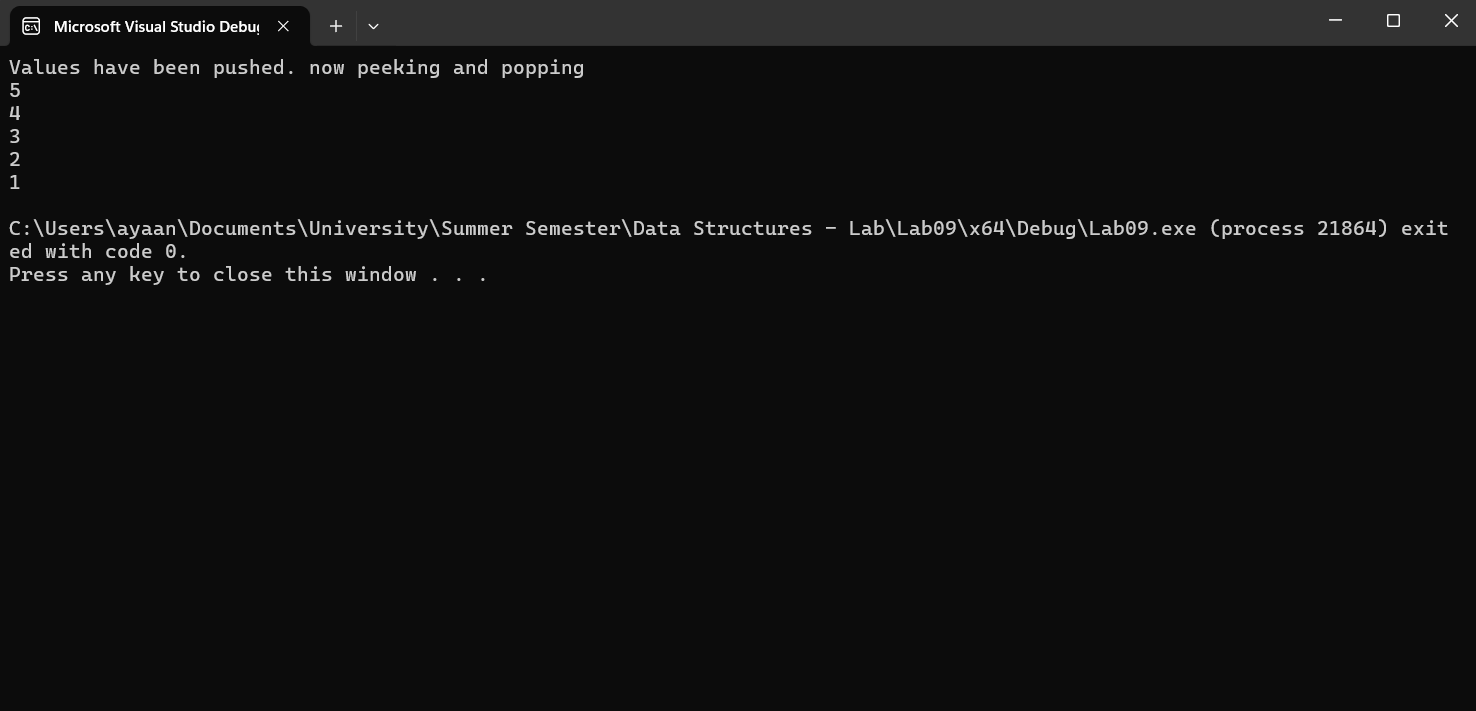
Task 01:



#include<iostream>

using namespace std;

class Stack

{

public:

int arr[1000];

int top;

Stack()

{

top = -1;

//for (int i = 0; i < 1000; i++)

//{

// arr[i] = -1;

//}

}

bool isEmpty()

{

if (top <= -1)

{

return 1;

}

return 0;

}

void push(int x)

{

if (top >= 999)

{

cout << "Stack full\n";

}

top++;

arr[top] = x;

}

int pop()

{

if (!isEmpty())

{

int temp = arr[top];

top--;

return temp;

}

}

int peek()

{

if (!isEmpty())

{

cout << arr[top] << endl;

return arr[top];

}

cout << "Stack is empty\n";

}

};

int main()

{

Stack stack;

stack.push(1);

stack.push(2);

stack.push(3);

stack.push(4);

stack.push(5);

cout << "Values have been pushed. now peeking and popping\n";

while (!stack.isEmpty())

{

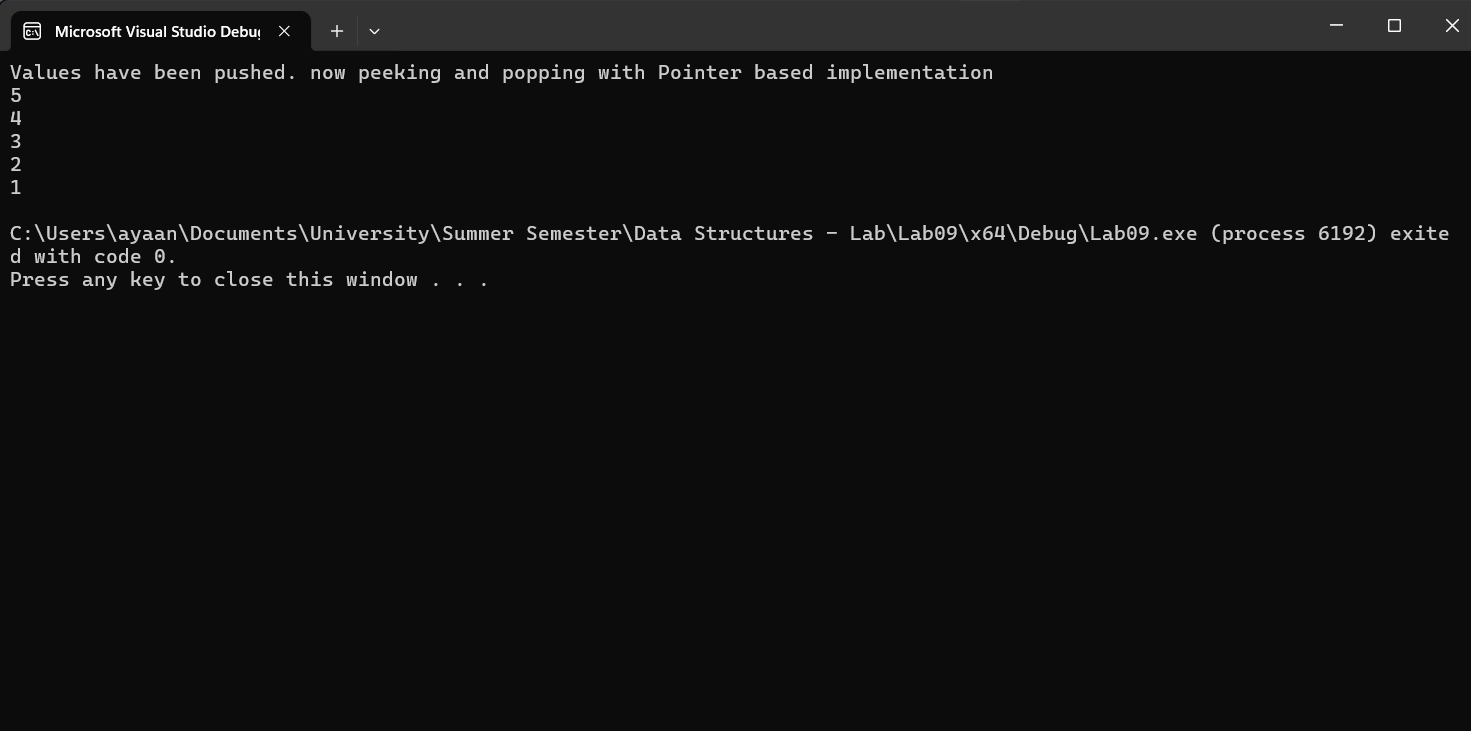
stack.peek();

stack.pop();

}

}

Task 02:



template<class T>

class Node

{

public:

T data;

Node\* next;

Node()

{

data = -1;

next = NULL;

}

Node(T x, Node\* y)

{

data = x;

next = y;

}

};

template<class T>

class Stack

{

public:

Node<T>\* top;

Node<T>\* current;

Stack()

{

top = NULL;

}

bool isEmpty()

{

if (top == NULL)

return 1;

return 0;

}

void push(const T DataItem)

{

Node<T>\* ptr = new Node<T>;

Node<T>\* ptr2 = new Node<T>;

ptr->data = DataItem;

ptr2 = top;

top = ptr;

top->next = ptr2;

}

T peek()

{

if (isEmpty())

return -1;

return top->data;

}

void pop()

{

Node<T>\* ptr = new Node<T>;

ptr = top;

top = top->next;

delete ptr;

}

void clear()

{

while (!isEmpty())

{

pop();

}

}

~Stack()

{

clear();

}

};

int main()

{

Stack<int> stack;

stack.push(1);

stack.push(2);

stack.push(3);

stack.push(4);

stack.push(5);

cout << "Values have been pushed. now peeking and popping with Pointer based implementation\n";

while (!stack.isEmpty())

{

cout << stack.peek() << endl;

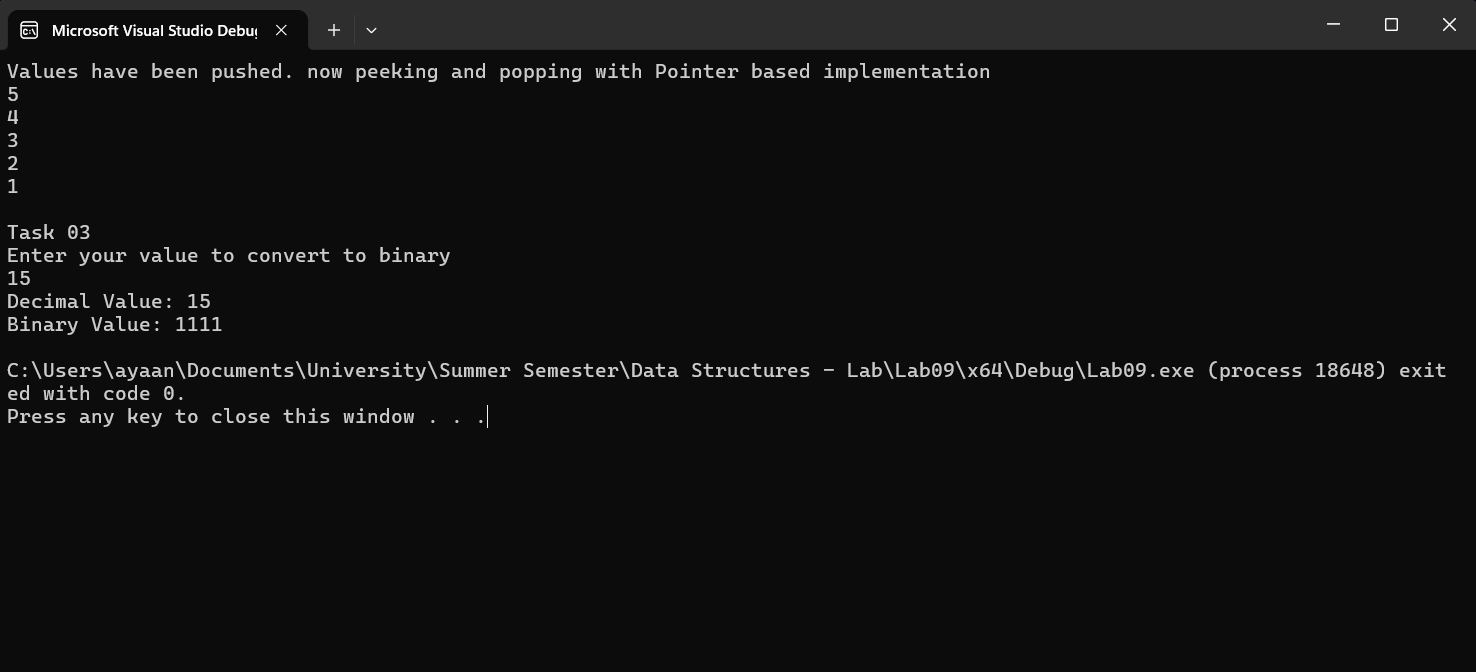
stack.pop();

}

return 0;

}

Task 03:



string convertDecToBin(int val1)

{

Stack<int> stack2;

int temp = val1;

string final;

int val2;

while (temp >= 1)

{

val2 = temp % 2;

stack2.push(val2);

temp /= 2;

}

while (!stack2.isEmpty())

{

final.append(std::to\_string(stack2.peek()));

stack2.pop();

}

return final;

}