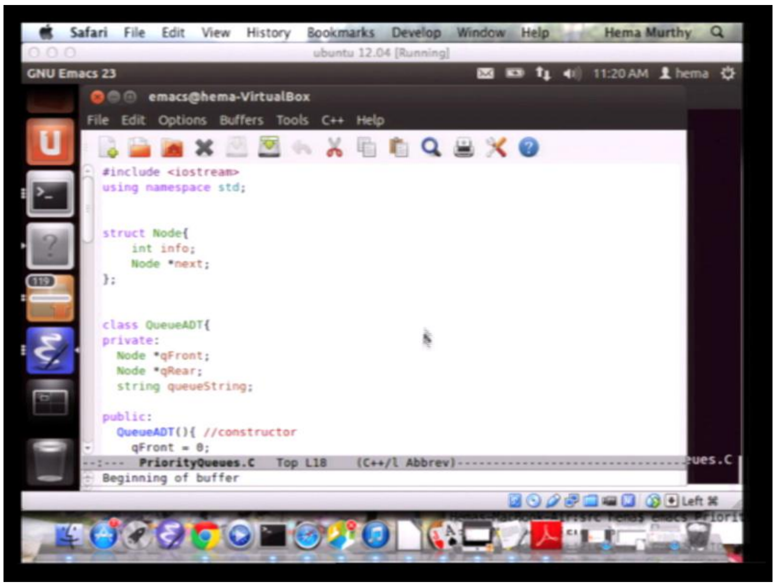
Programming, Data Structures and Algorithms
Prof. Hema A Murthy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 54
Assignment on Data Structures

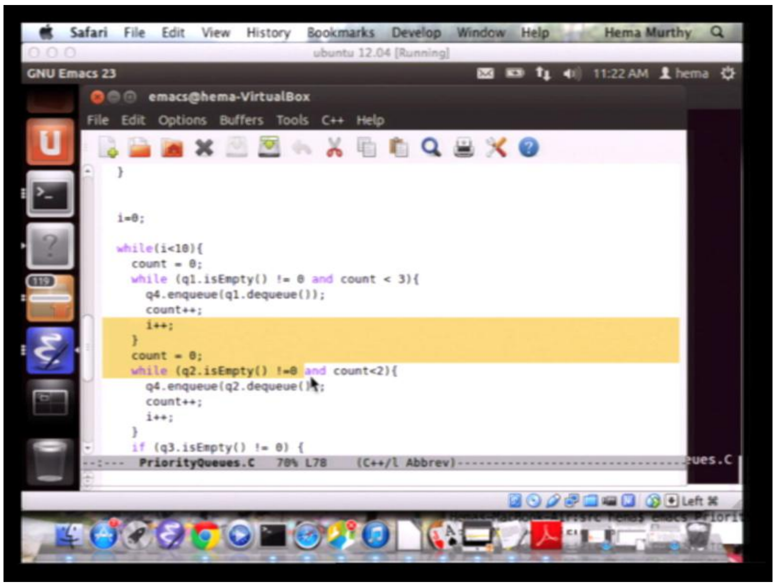
(Refer Slide Time: 00:09)


Safari File Edit View History Bookmarks Develop Window Help
Hema Murthy
ubuntu 12.04 [Running]
GNU Emacs 23
11:20 AM hema
emacs@hema-VirtualBox
File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;


struct Node{
    int info;
    Node *next;
};


class QueueADT{
private:
  Node *qFront;
  Node *qRear;
  string queueString;

public:
  QueueADT(){ //constructor
    qFront = 0;
--:---  PriorityQueues.C   Top L18    (C++/l Abbrev)----------------------------
Beginning of buffer
```

What we going to do is, this is the solution to the queues problem, what where you given, you have told that, that you should implement use the queue ADT that is given. And basically what we are looking at is, you are expected to create three different queues.

(Refer Slide Time: 00:27)



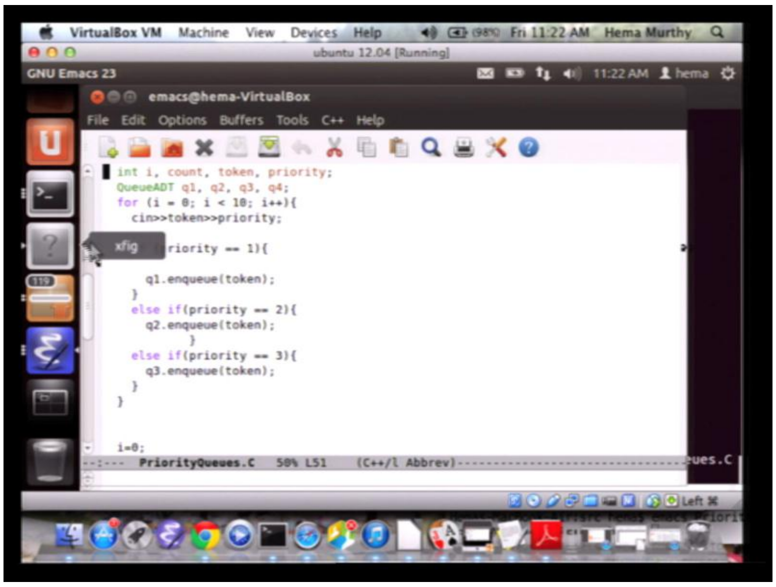
```
  }


  i=0;

  while(i<10){
    count = 0;
    while (q1.isEmpty() != 0 and count < 3){
      q4.enqueue(q1.dequeue());
      count++;
      i++;
    }
    count = 0;
    while (q2.isEmpty() !=0 and count<2){
      q4.enqueue(q2.dequeue());
      count++;
      i++;
    }
    if (q3.isEmpty() != 0) {
--:---  PriorityQueues.C   70% L78    (C++/l Abbrev)----------------------------------
```

So, basically it is like doctor's clinic and there are severe cases and there are non-severe cases and there are moderate cases, for what is that is three different queues are created. Obviously, the people with severe problems have to be treated in greater numbers than the moderate people and the persons with not any severity at all can be treated the least. But, the same time, but you cannot do is even many severe people you cannot only treat severe people and not treat the people with moderate illness and people with very less illness.

So, otherwise people will get tired they will not go to the doctor at all. So, what was suggested in this was, you create a fourth queue to the doctor. So, let us say there was a receptionist at the desk where you go and register your name, the person listens to your complaints and realize whether it is severe, moderate or what you called the basically what it does is, it looks at your priority is your priority 1 is your priority 2.

Basically depending upon the severity and depending upon the priority it is supports to put you in once the person identifies what kind of severe diseases that you have, deputy if your priority is let us say very high, when you that is severe goes into the first queue and if your priority is not so, severe goes into the second queue and that is exactly what is done.

(Refer Slide Time: 02:08)

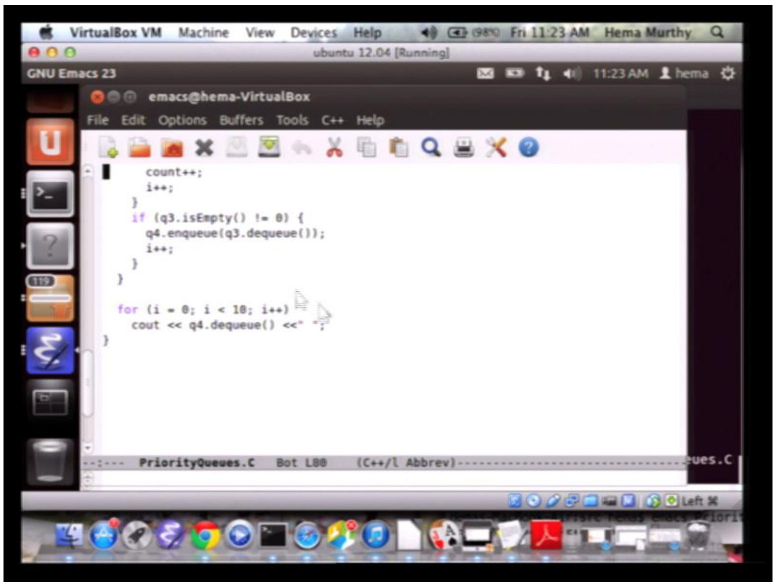


```
int i, count, token, priority;
QueueADT q1, q2, q3, q4;
for (i = 0; i < 10; i++){
  cin>>token>>priority;

  riority == 1){

    q1.enqueue(token);
  }
  else if(priority == 2){
    q2.enqueue(token);
        }
  else if(priority == 3){
    q3.enqueue(token);
  }
}


i=0;
--:---  PriorityQueues.C   50% L51    (C++/l Abbrev)--------
```

So, basically what we doing is we taking as input at token and priority, if priority is equal to 1 it is put into the first queue, priority is equal to 2, it is put into the second queue moderate and priority is the not so, severe, not at all severe, you go into that third queue. And what is the question say it say that is doctors queue. And what we should ensure is that, there are 10 patients in the queue doctors queue for example, at any point time here all having 10 elements in the queue.

So, what you do is the way you allot them is that, first you put 3 patients from the high priority queue, when you put 2 patients severe there from the moderate queue and then put 1 patient from the least moderate queue and you keep repeating this order. So, that for example, if there are let us say 6 people who have severe problem, then the first three will get served, then the moderate person will get served, two moderate people will get served and then one person will non-severity will be served and again the three people will severity will be served, that is the application.

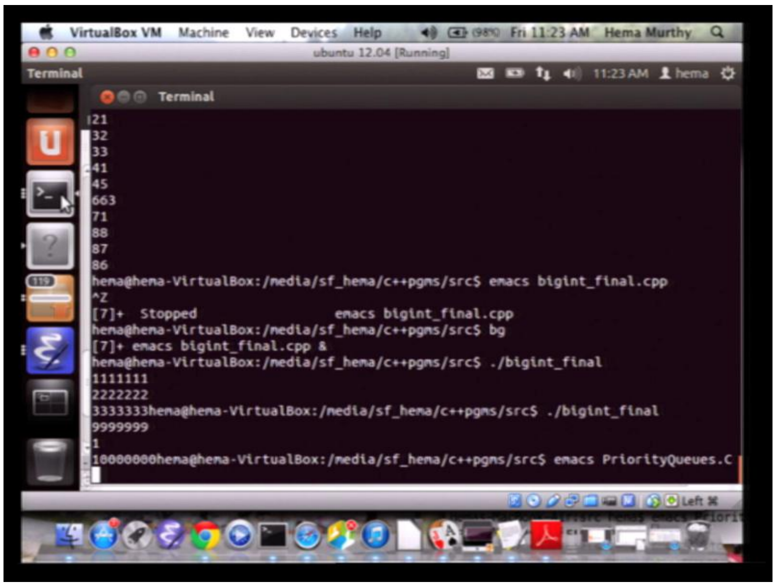
(Refer Slide Time: 03:19)



```
      count++;
      i++;
    }
    if (q3.isEmpty() != 0) {
      q4.enqueue(q3.dequeue());
      i++;
    }
  }

  for (i = 0; i < 10; i++)
    cout << q4.dequeue() <<" ";
}
```

And what are you supposed to give, is supposed to output the elements in the queue. So, let us look at this program as it runs.

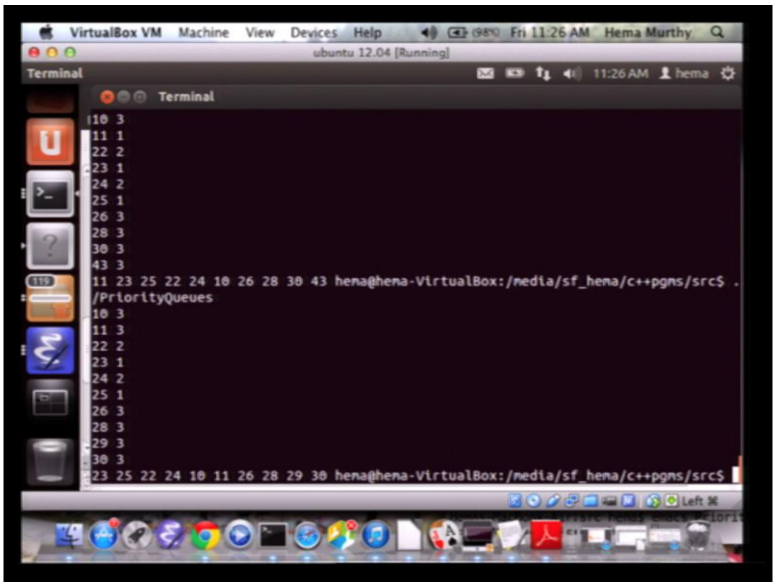
(Refer Slide Time: 03:24)



```
21
32
33
41
45
663
71
88
87
86
hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ emacs bigint_final.cpp
^Z
[7]+  Stopped                 emacs bigint_final.cpp
hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ bg
[7]+ emacs bigint_final.cpp &
hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ ./bigint_final
1111111
2222222
3333333hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ ./bigint_final
9999999
1
10000000hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ emacs PriorityQueues.C
```

So, very, very simple it is problem on queues which you already studied is not really priority queue. What we are doing is you remember that queue that we studied in the class for example. Most important point here is that, if you look at queue 1, queue 2 and queue 3 people enter the queue what is happening is, first comes first serve we are, in the sense for that particular list. That is will a particular severity if you come, you going into the particular list, even the next person who comes into with the particular severity, when you go into the same list,, but it end of in that queue that is exactly what we are having over here.

(Refer Slide Time: 04:05)



```
10 3
11 1
22 2
23 1
24 2
25 1
26 3
28 3
30 3
43 3
11 23 25 22 24 10 26 28 30 43 hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ .
/PriorityQueues
10 3
11 3
22 2
23 1
24 2
25 1
26 3
28 3
29 3
30 3
23 25 22 24 10 11 26 28 29 30 hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$
```

So, let us run this, what are you given, you given a token and priority. So, let say 10 3, 11 1, 22 2, 23 1, 24 2, 25 1, 26 3, 28 3 and at say 30 3 and 43 3 because normally we will have people with low priority. Now, what is it giving so, number 1 means token and priority here. So, basically if you notice the 11, 23 and 25 3 patients who are having severe problems.

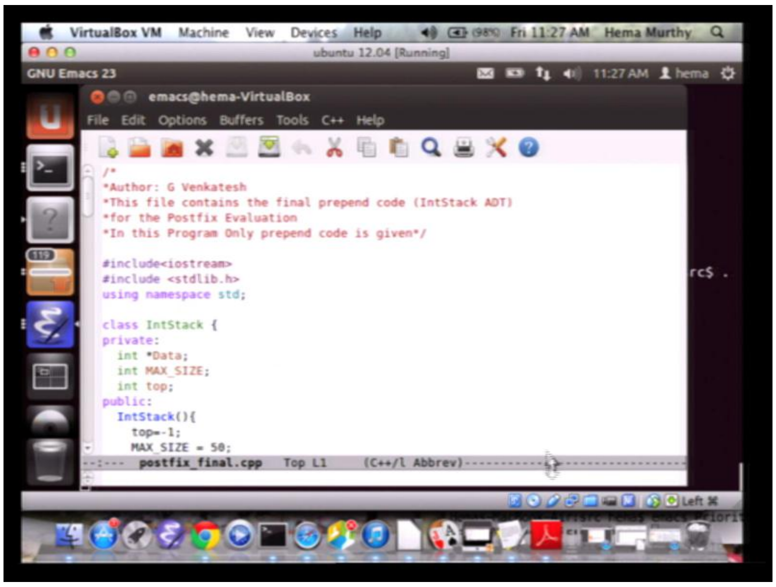
So, doctor's queue has 11, 23and 25 the token number of this paper and then what happens, there are two people with a lower severity 22 and 24. So, the next 2 guys go over here and then what do we have, the rest of them. Because, there are ideally you are supposed to allow only one person with low severity, but since there are no other patients with higher severity, it goes into this particular list.

Let us run this program again, let say we have 10 3, 11 3, 22 2, 23 1, 24 2, 25 1, 26 3, 28 3, and 29 3, 30 3, now what happens here. So, you have basically we still have a six people over here 1, 2, 3, 4, 5, 6 that is correct we have basically two people. So, 23 and 25 are high priority we here then. So, still nobody else with higher priority 22 and 24 come and finally, all the other people.

So, this is about using the queue ADT, because defined and using it to create three different queues, remember this is not a priority queue ((Refer Time: 06:45)). It say that, every queue is a first in first out queue,, but there are three different queues,, but it creates what is called a fourth queue which corresponds to the queue of the doctor, in which we put elements into that queue depending upon the priority. So, this is something that you need to understand.

And this shows the execution of the program and what are we showing here, we are giving the token numbers of the elements in the doctors queue. So, now we look at a solution to another problem, which is the postfix evaluation problem.

(Refer Slide Time: 07:31)



VirtualBox VM Machine View Devices Help
ubuntu 12.04 [Running]
GNU Emacs 23
emacs@hema-VirtualBox
File Edit Options Buffers Tools C++ Help

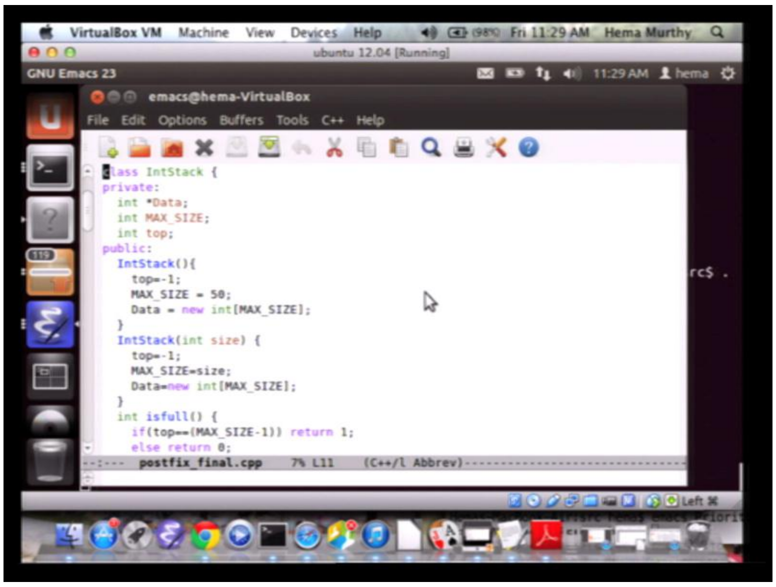
```
/*
*Author: G Venkatesh
*This file contains the final prepend code (IntStack ADT)
*for the Postfix Evaluation
*In this Program Only prepend code is given*/

#include<iostream>
#include <stdlib.h>
using namespace std;

class IntStack {
private:
  int *Data;
  int MAX_SIZE;
  int top;
public:
  IntStack(){
    top=-1;
    MAX_SIZE = 50;
```

```
--:---  postfix_final.cpp   Top L1    (C++/l Abbrev)------------
```

We always saw the infix to postfix conversion. So, let us look at this postfix evaluation problem, this uses a stack and here remember, when we did the infix to postfix conversion, we were given an expression and we said the expression has to be converted to postfix. So, when we did that what we did was, we defined or stack of characters if we remember I of some particular element I.

And in that case what we did was, why we define a stack of characters. Because, in that problem we put the operators onto the stack and as soon as you came across an operator, which has lower priority then what was there on the stack lower or equal to priority, we popped all the elements of the stack.

(Refer Slide Time: 08:17)

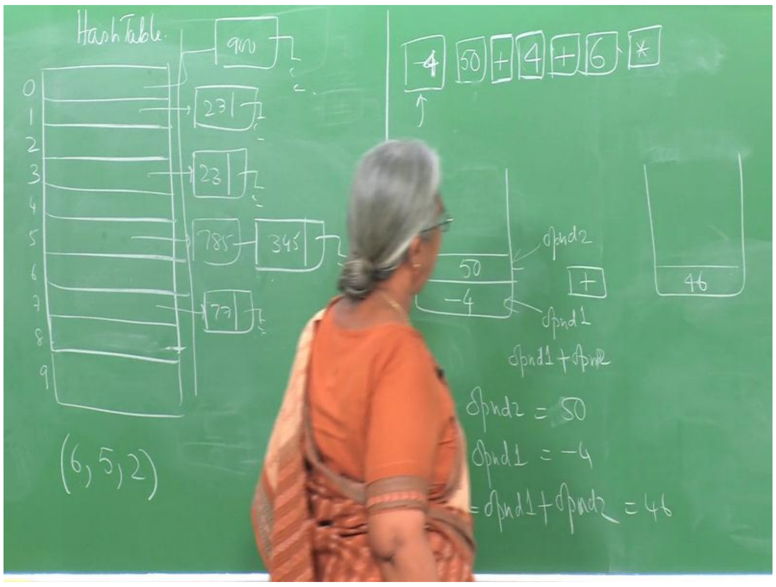


```
class IntStack {
private:
  int *Data;
  int MAX_SIZE;
  int top;
public:
  IntStack(){
    top=-1;
    MAX_SIZE = 50;
    Data = new int[MAX_SIZE];
  }
  IntStack(int size) {
    top=-1;
    MAX_SIZE=size;
    Data=new int[MAX_SIZE];
  }
  int isfull() {
    if(top==(MAX_SIZE-1)) return 1;
    else return 0;
```

```
--:---  postfix_final.cpp     7% L11     (C++/l Abbrev)-------------------
```

Now, this is slightly different problem, you are given opposed fix this expression and what you suppose to do is let me take this example over here. So, what we mean by this is the following in the postfix evaluation. Let us look at this problem here,. So, you want to add some integers in the postfix evaluation problem.

(Refer Slide Time: 08:54)

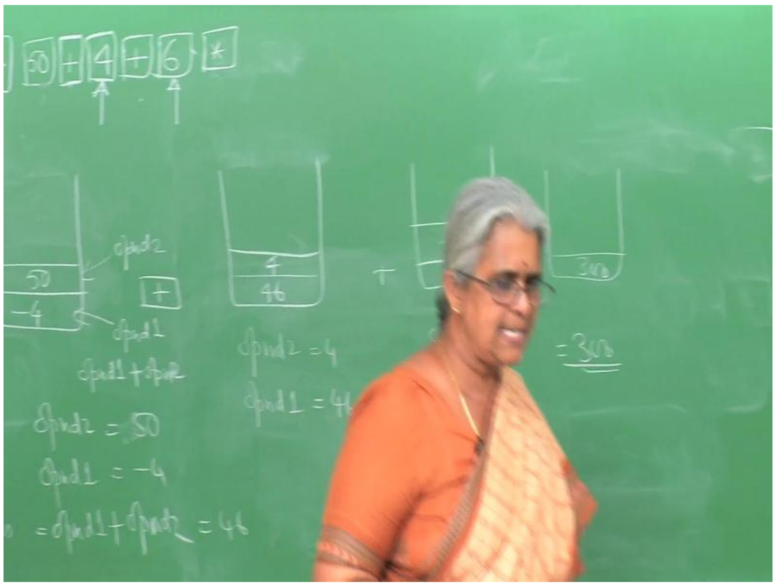


So, let us say I have the numbers minus 4 50 plus and this is one token and one token and let us it 4 plus 6 and star. So, this is a postfix expression,. So, now, , I am putting this boxes around all of this, because I cannot treat them as simply as a string of stream of characters for the simple reason that the integers. For example, where you look at that in terms of characters, depending upon the number of digits that they have can have more than one digit like this example over here.

So, what are we excepted to do in this, we notice that this minus 4 is a number is a unary number minus over here. So, as given as a number here,. So, I want minus 4 plus,. So, as soon as I,. So, what are we do is, I want to use a stack to do this problem. So, what do we do here now, how do I do this now the way it works like this, as soon as you seen an operand you put it onto the stack. Now, minus 4 is a operand and put it onto the stack 50 is the next operand we put it onto the stack.

But, as soon as you see the operator what you do is, this is operand 2 and this is operand 1 and you perform operand 1 plus operand 2. So, what we do is we first pop of both these elements. So, let say I put opnd 2 first element has been popped is 50 and the second element has been popped opnd 1 is minus 4, when you perform opnd 1 plus because we have seen and the operator here opnd 2 and then minus 4 plus 50 is 46 let say this is the result and this is equal to 46. So, since we have popped both the elements, the stack is again empty I push 46 back of the stack.

(Refer Slide Time: 11:24)



Then what you do, we have now this is done come to the next token at push it onto the stack, because it is an operant and again I see a plus. Therefore, again now opnd 2 it will to 4 opnd 1 equal to 46 and the result is,. So, a pop both of them out. So, again my stack is empty, when add these two numbers and in push it back onto the stack. So, push that back onto the stack, then what do I do next stack here we have already finish the operator here at push it on to the stack, then I see multiplication now what do I expect.

So, basically the opnd 2 6 opnd 1 is 50 and we do multiplied both of them result is 300 push it back onto the stack, now you come to the end of the expression. So, I pop the element of the stack and that is the result. So, let us exactly what is being done in this part of the problem. So, what is being done is here, if you notice there is in stack of particular size, there is an implemented and we push the element onto the stack.

But, remember that what is this element now, the element is that are being push onto the stack or a type integer, remember this integer stack. But, what are we doing when we are,. So, basically I here is something a result which will compute, depending upon the operator we will compute multiplication and what is that, as soon as we get the operands which are popped of the stack it compute the operation and returns the result.

Then, here is a function which converts a character string to an integer, there is see why this is required in a little byte. And basically you have to take your sign and. So, on and. So, forth, then remember we are looking at tokens there. So, it is looks for a token which is purely and an operator and that is what is being passed over here. Now, let us go to the main program here, what are we doing we are defining a character array called postfix, which consists of a string of integers. Why are we doing this, because remember that when you look at a postfix expression, there is a integer here, there is an integer here,, but there is an operator here. But, you going to give this string as it is,, but as we already saw as you explained sometime back curly each element in the postfix string is not a single digit number, it can have multiple digit. So, what we do is, we represents each one of the tokens, we call this tokens in the postfix expression, this token 1 token 2, 3, 4, 5, 6 and 7 each one of them as a string.

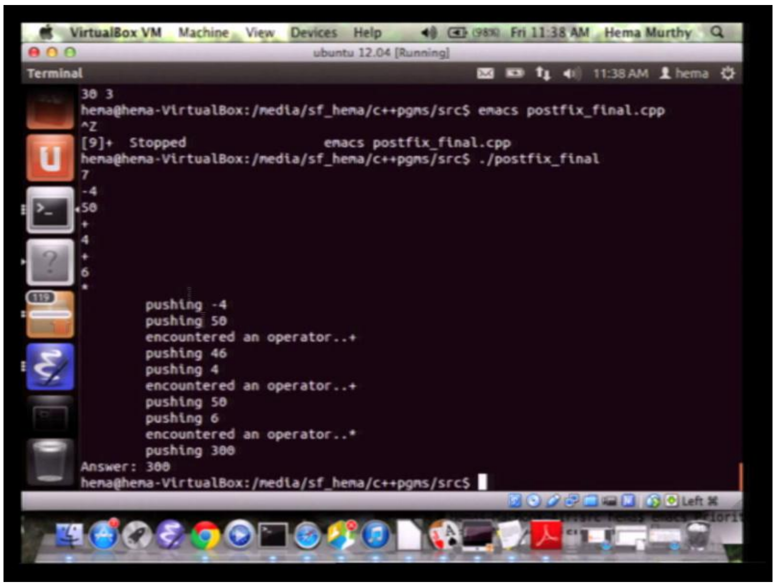
So, to do that what is done is a character array of n, suppose first what we do is you get the number of tokens in the string. So, what it is we are saying here, how many token are there 1, 2, 3, 4, 5, 6 and 7,. So, 7 tokens in the string. So, what it does here, you here for example, the input n is given and then it create a number of such character strings which are of size n and then what it is do, we create assumes that the string is no more than 5 characters long and then what it does it.

So, there is an allocation that is done here the string is read and it is put into this postfix array. So, what is the postfix array is actually a two dimensional array and what is it having, if you look at a two dimensional array in the first element is 4, second element is 50, that is each element of the array is a string of length 5 characters maximum plus 6 and star in this particular example. So, this is the postfix array and then what is done, you once you have done that it is basically going to call, then what is happening here we need to perform this evaluation.

So, what is done is it look at what the,. So, what is doing here it gets each one of the elements as I said that there are n tokens and for each one of the token it is check whether is an operator or not. Otherwise, it pops the elements from the operator for example, v 2 and v 1 this is operand 2, this is operand 1 and then performs the operation, then what this do once it evaluate it is evaluation is the result here, we pushes it back onto the stack.

So, where encountering and operand for example, because it is also string it converts the character string to an integer and pushes it and then pushes that onto the stack. Because, the stack is a stack of integers, because we do not need anything other than the stack of integers, because we putting the operands onto the stack. So, let see how this program runs.

(Refer Slide Time: 17:10)



```
30 3
hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ emacs postfix_final.cpp
^Z
[9]+  Stopped                 emacs postfix_final.cpp
hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ ./postfix_final
7
-4
50
+
4
+
6
*
        pushing -4
        pushing 50
        encountered an operator..+
        pushing 46
        pushing 4
        encountered an operator..+
        pushing 50
        pushing 6
        encountered an operator..*
        pushing 300
Answer: 300
hema@hema-VirtualBox:/media/sf_hema/c++pgms/src$ 
```

So, postfix final I to give 7 because I am let say I looked at 7 operands here minus 450 plus 4 plus 6 and star. So, what it is doing it pushing minus 4 onto the stack, pushing 50 onto the stack, it encounter an operator next as indicated here. Then it gets the result 46 it is pushes it onto the stack, there is a season operand again pushes the operand onto the stack. Again it encounters an operator then what it is do, it computes the sum of pops both 46 and 4 from the stack ask them gets the result 50 pushes it back onto the stack.

Then, it comes encounter an operand again 6 it pushes it back onto the stack then it encounters an operator and what it is do, it pushes once again encounters an operator is going to pop 50 and 60 6 first and then 50 and then perform the operation of multiplication, the result is 300 and it pushes 300 back onto the stack.

But, clearly by now we are come off to the end of expression and has been exhausted the 7 therefore, it is give you the result. So, this is how the postfix evaluation program can be implemented for those of few have perhaps not done it properly, I would encourage for the go back and try it out again.