G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)



**Report on Operating Systems for Simulation based CA**

**Course Code:** CSE 316

**Course Name:** OPERATING SYSTEMS

**Submitted To:** Geetika Chatley mam

**Submitted by:** G. Pradeep Reddy

**Registration no:** 11702919

**Roll no:** 21

**Email Address:** [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

**Github Link:** https://github.com/pradeep0825/Pradeep0825.git

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

# Code:

**Solution code assigned to you**

#include <stdio.h> #define capacity 20 int event[capacity];

int eventblocked[capacity]; int eventId=0;

int doeventopen(){

//creates a new event, returning eventId on success, -1 on failure if(eventId!=capacity)

{

event[eventId]=1; eventId++;

return eventId;

}

else

return -1;

}

int doeventclose(int eventId){

//Destroy the event with the given eventId and signal any processes waiting on the event to leave the event. Return number of processes signalled on success and -1 on failure.

if(event[eventId]==1)

{

event[eventId]=0;

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

return eventId--;

}

else

return -1;

}

int doeventwait(int eventId){

//Blocks the processes until the event is signalled. Return 1 on success and -1 on failure. if(eventId<=capacity&&eventId>=0)

{

eventblocked[eventId]=1; return 1;

}

else

return -1;

}

int doeventsig(int eventId){

//Unblocks all waiting processes; ignored if no processes are blocked. Return number of processes signalled on success and -1 on failure.

if(eventId<=capacity&&eventId>=0)

{

eventblocked[eventId]=0; return 1;

}else

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

return -1;

}

int main(){

for(int i=0;i<23;i++)

{

int k=doeventopen(); if(k!=-1) //Success

{

printf("Process created successfully with event ID : %d\n",k);

}

else //failure

{

printf("Process failed to create\n");

}

}

for(int i=1;i<24;i++)

{

int k=doeventclose(i); if(k!=-1) //Success

{

printf("Process closed successfully with event ID : %d\n",k);

}

else //failure

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

{

printf("Process failed to close\n");

}

}

for(int i=1;i<23;i++)

{

int k=doeventwait(i);

if(k!=-1) //blocked event successfully

{

printf("process blocked successfully\n");

}

else //failed to block the process printf("process failed to block\n");

}

for(int i=1;i<23;i++)

{

int k=doeventsig(i);

if(k!=-1) //Unblocked event successfully

{

printf("process unblocked successfully\n");

}

else //failed to unblock the process printf("process failed to unblock\n");

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

}

int n,m; printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); printf("Please enter your choice:\n"); printf("1. Create new event\n");

printf("2. Close the event with eventID\n"); printf("3. Block an event with eventID\n"); printf("4. Unblock an event with eventID\n\n"); printf("5. Check whether an event is closed\n"); printf("6. Check if an event is blocked\n"); printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n"); scanf("%d",&n);

switch(n)

{

case 1: if(doeventopen()!=1) //Success

{

printf("Process created successfully\n");

}

else //failure

{

printf("Process failed to create\n");

}

break;

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

case 2:

printf("Please enter eventID(1 to 20):\n"); scanf("%d",&m);

if(m<1||m>20){

printf("I already said eventID should be between 1 to 20\n");break;} if(doeventclose(m)!=-1) //Success

{

printf("Process closed successfully\n");

}

else //failure

{

printf("Process failed to close\n");

printf("Try opening an event before closing it!\n");

}

break;

case 3:

printf("Please enter eventID(1 to 20):\n"); scanf("%d",&m);

if(m<1||m>20){

printf("I already said eventID should be between 1 to 20\n");break;} if(doeventwait(m)!=-1) //blocked event successfully

{

printf("process blocked successfully\n");

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

}

else //failed to block the process printf("process failed to block\n");

break;

case 4:

printf("Please enter eventID(1 to 20):\n"); scanf("%d",&m);

if(m<1||m>20){

printf("I already said eventID should be between 1 to 20\n");break;} if(doeventsig(m)!=-1) //Unblocked event successfully

{

printf("process unblocked successfully\n");

}

else //failed to unblock the process printf("process failed to unblock\n"); break;

case 5:

printf("Please enter eventID(1 to 20):\n"); scanf("%d",&m);

if(m<1||m>20){

printf("I already said eventID should be between 1 to 20\n");break;} if(event[m]!=1)

{

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

printf("No the event is not closed!\n");

}

else

printf("Yes the event is closed\n"); break;

case 6:

printf("Please enter eventID(1 to 20):\n"); scanf("%d",&m);

if(m<1||m>20){

printf("I already said eventID should be between 1 to 20\n");break;} if(eventblocked[m]!=1)

{

printf("No the event is not blocked!\n");

}

else

printf("Yes the event is blocked!\n"); break;

}

return 0;

}

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

# Explain the problem in terms of Operating system concept?

**Description:** The problem given is to implement four methods in c language. In terms of operating system, process synchronization is the one word which can be suited to describe the problem.

# Write the algorithm for proposed solution of the assigned problem? Algorithms:

**Algorithm 1 (int doeventopen()) :**

Step 1: if eventId not equal to capacity

Step 2 : Then event[eventId]=1 and eventId++ and return eventId Step 3: else return -1

# Algorithm 2 (int doeventclose(int eventId) ) :

Step 1: if event[eventId] is equal to 1

Step 2: then event[eventId]=0 and return eventId Step 3: else return -1

# Algorithm 3 (int doeventwait(int eventId)) :

Step 1: if eventId is less than or equal to capacity and eventId is greater than or equal to zero Step 2: then eventblocked[eventId]=1 and return 1

Step 3: else return-1

# Algorithm 4 (int doeventsig(int eventId)) :

Step 1: if eventId is less than or equal to capacity and eventId is greater than or equal to zero Step 2: then eventblocked[eventId]=0 and return 1

Step 3: else return -1

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

1. **Calculate the complexity of proposed algorithm?** For each line, the time complexity is constant or O(1) . But for testing, since I have used for loop it is O(n).

Although it looks as less complexity but my code has space complexity.

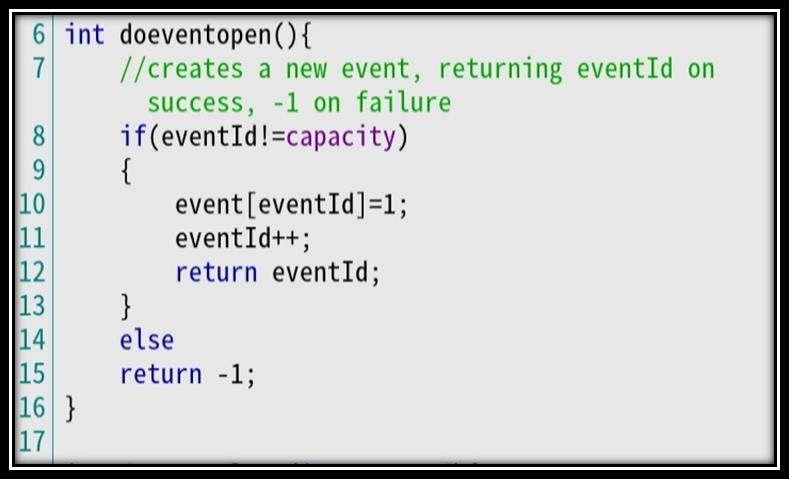
This is mainly because I have used two arrays one is for events and the other one for flags which represent either the process is busy or free.

I thought of using linked list, but in linked list for travelling to the nth node requires n steps whereas in arrays we can directly use indexing ( for example arr[n] will get the required item)

Primary reason behind using arrays is to reduce time complexity and make the code easier to implement and debug.

# Explain all the constraints given in the problem. Attach the code snippet of implemented constraint?

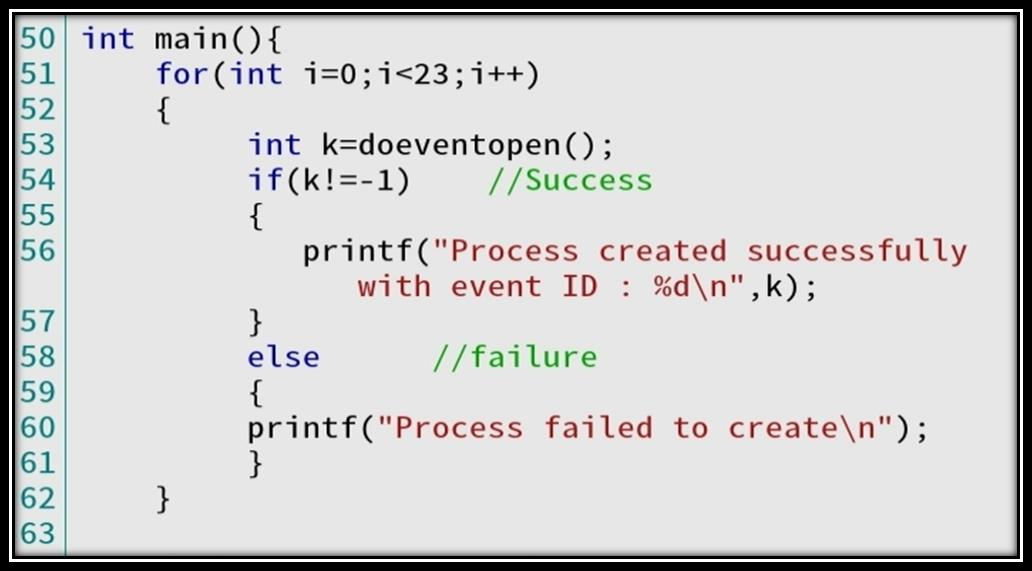
First constraint: int doeventopen(): creates a new event, returning eventId on success, -1 on failure



# Code snippet:

**Code snippet for testing:**

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)



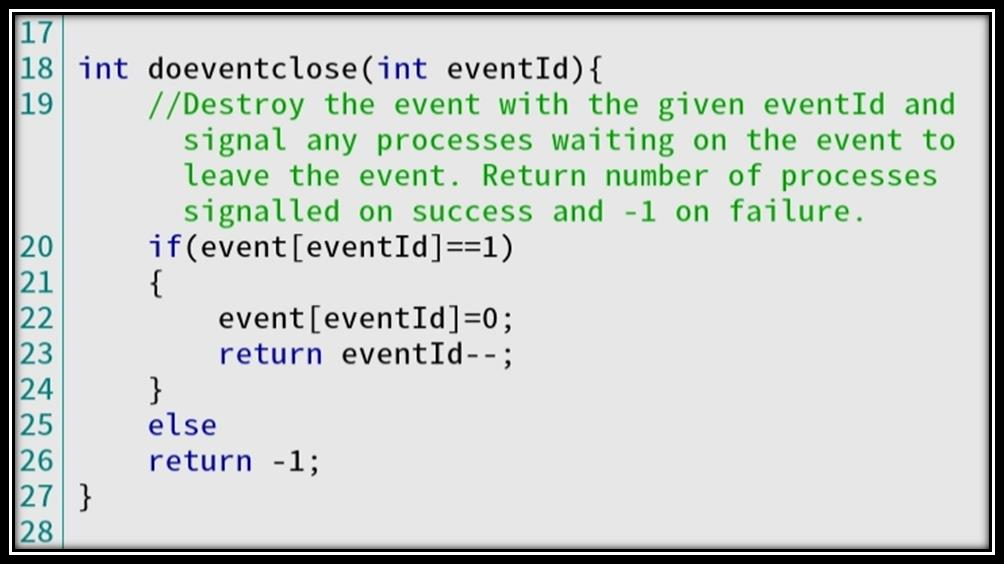
# Code spinnet for results:



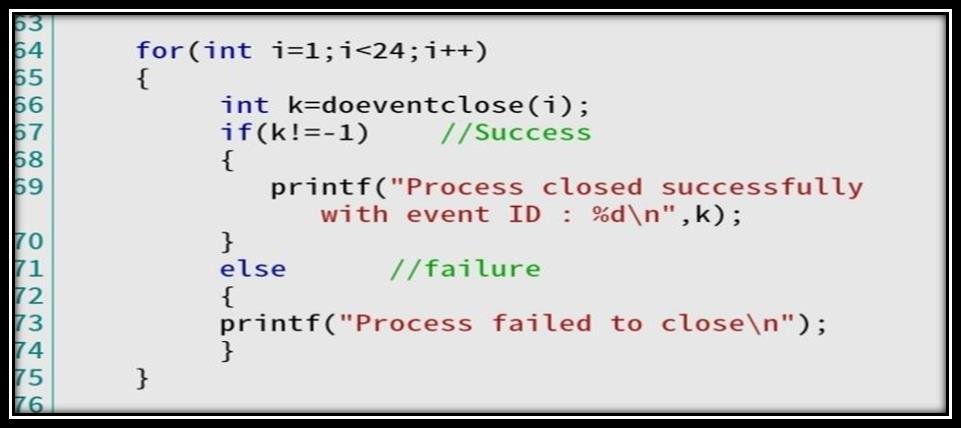
Second constraint: int doeventclose(int eventId): Destroy the event with the given eventId and signal any processes waiting on the event to leave the event. Return number of processes signalled on success and -1 on failure.

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

# Code Snippet:

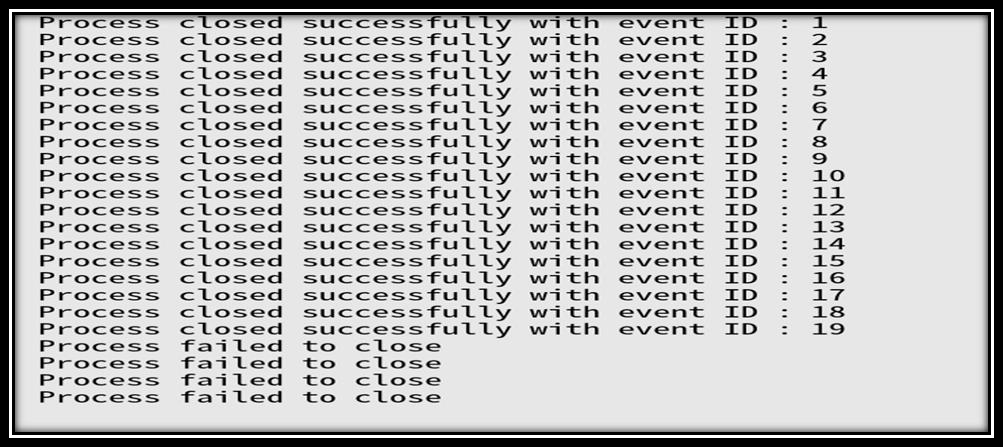


**Code snippet for testing:**



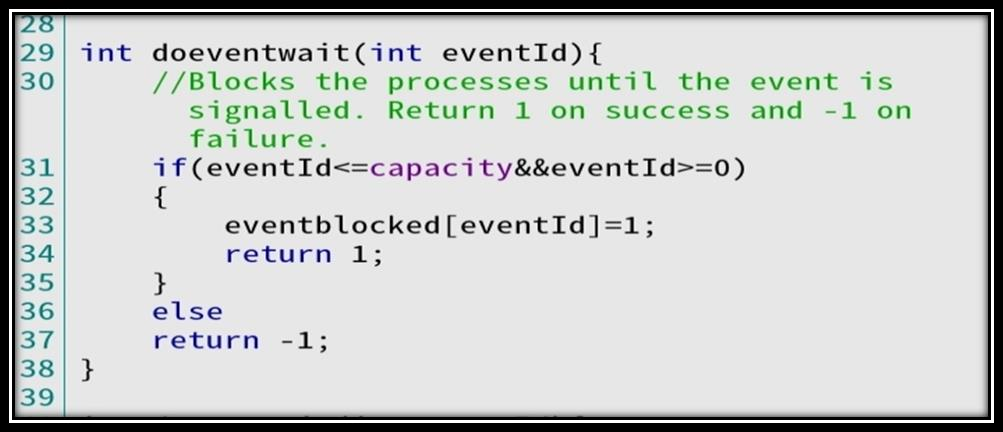
G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

# Code Snippet for results:



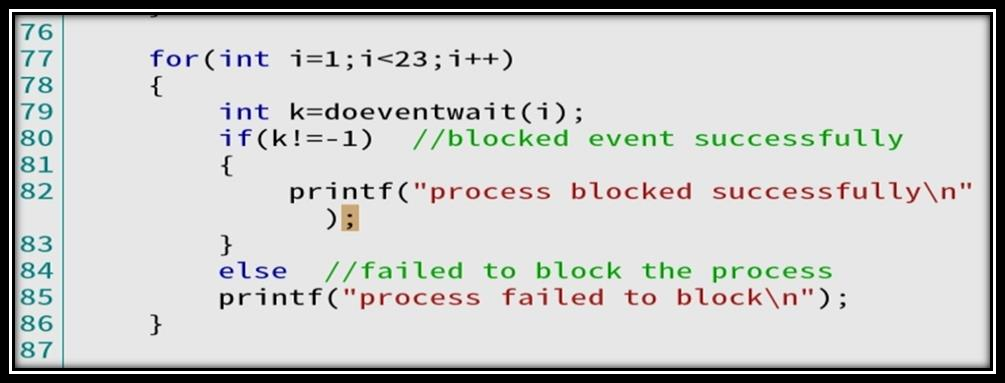
**Third constraint:** int doeventwait(int eventId) :Blocks the processes until the event is signalled. Return 1 on success and -1 on failure.

# Code snippet:



**Code snippet for testing:**

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)



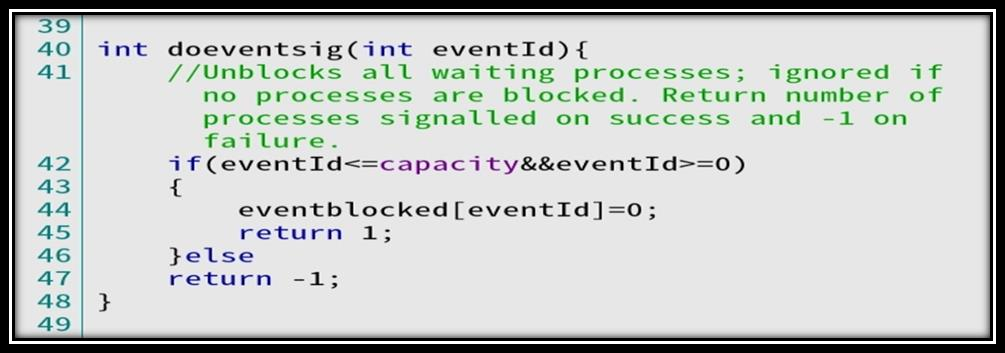
# Code snippet for results:



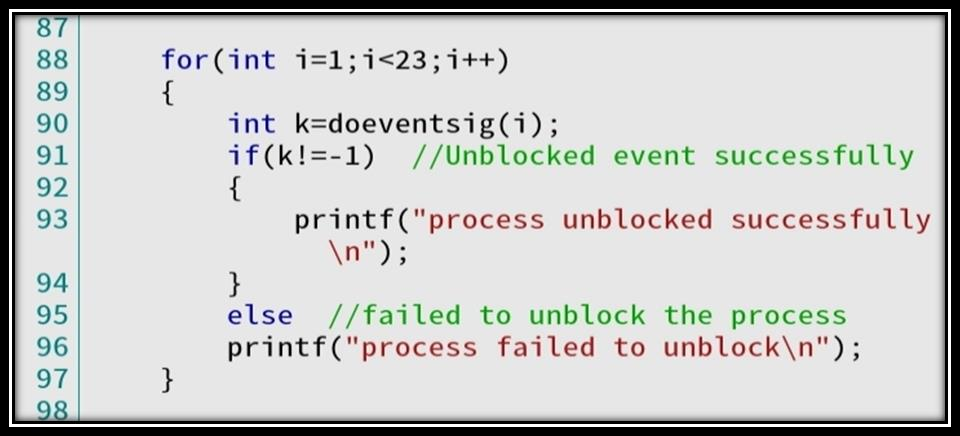
**Constraint 4:** int doeventsig(int eventId): Unblocks all waiting processes; ignored if no processes are blocked. Return number of processes signalled on success and -1 on failure.

G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

# Code snippet:



**Code snippet for testing:**

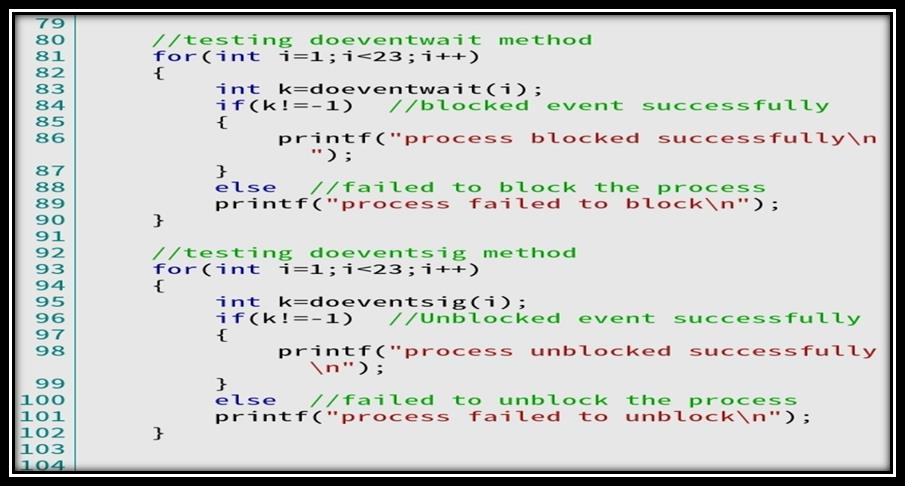


**Code snippet for results:**



16 | P a g e

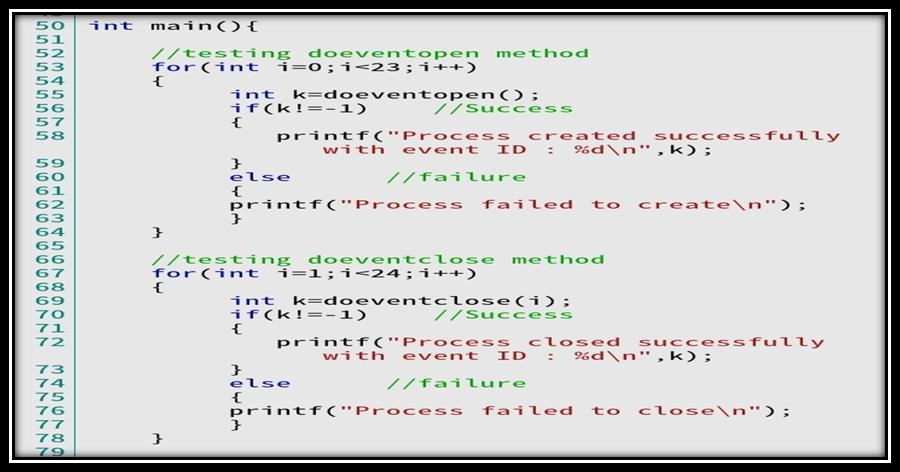
G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)



17 | P a g e

# If you have implemented any additional algorithm to support the solution, explain the need and usage of the same?

Yes, for testing purpose I have used four for loops each one to test the given c methods.



G.Pradeep Reddy 11702919 [Pradeeppradeep0825@gmail.com](mailto:Pradeeppradeep0825@gmail.com)

# Explain the boundary conditions of the implemented code?

One condition is the array index may go out of bounds. This may lead to runtime error or unexpected behaviour. To prevent this before adding elements I have checked whether the index is below the capacity.

# Explain all the test cases applied on the solution of the assigned problem?

I have used four for loops each one to test each of the four methods.

1. **Have you made minimum five revisions on GitHub? Github Link:** https://github.com/pradeep0825/Pradeep0825.git

18 | P a g e