**OS : Practical Assignment no. 7**

***Title* :**

Write a program for implementing Page Replacement - FIFO, LRU, Optimal.

***Program* :**

#include<bits/stdc++.h>

#include<iostream>

using namespace std;

int present(int table\_frame[], int nf, int page)

{

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

{

if(page == table\_frame[i])

return 1;

}

return 0;

}

void printtable(int table\_frame[], int nf)

{

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

{

if(table\_frame[i] == -1)

cout<<" -- ";

else

cout<<" "<<table\_frame[i];

}

cout<<" || ";

}

int FIFO(int n, int nf, int table\_frame[], int pages[])

{

int count1=0,pos=0;

cout<<"\nPosition of frame table after each request : \n";

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

{

cout<<" "<<pages[i]<<" || ";

if(!present(table\_frame,nf,pages[i]))

{

table\_frame[pos] = pages[i];

pos = (pos+1)%nf ;//considering it as a queue

printtable(table\_frame,nf);

cout<<"Page Fault\n";

count1++;

continue;

}

printtable(table\_frame,nf);

cout<<endl;

}

return count1;

}

int findpos\_lru(int table\_frame[], int nf, int pages[], int curr, int np)

{

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

{

if(table\_frame[i] == -1)

return i;

}

int pos[nf] = {0};

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

{

pos[i] = -1e9;

for(int j=curr-1; j>=0; j--)

{

if(pages[j] == table\_frame[i])

{

pos[i] = j;

break;

}

}

}

int min1 = 1000000, retPos = -1;

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

{

if(min1 > pos[i])

{

min1 = pos[i];

retPos = i;

}

}

return retPos;

}

int LRU(int n, int nf, int table\_frame[],int pages[])

{

int count1=0;

cout<<"Position of frame table after each request : \n";

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

{

cout<<"Page table after request from "<<pages[i]<<" || ";

if(!present(table\_frame,nf,pages[i]))

{

int pos = findpos\_lru(table\_frame,nf,pages,i,n);

table\_frame[pos]=pages[i];

printtable(table\_frame,nf);

cout<<"Page Fault\n";

count1++;

continue;

}

printtable(table\_frame,nf);

cout<<"\n";

}

return count1;

}

int findpos\_optimal(int table\_frame[],int nf,int pages[],int curr,int np)

{

int i,j;

for(i=0;i<nf;i++)

{

if(table\_frame[i] == -1)

return i;

}

int pos[nf]={0};

for(i=0;i<nf;i++)

{

pos[i]=1e9;

for(j=curr+1;j<np;j++)

{

if(pages[j]==table\_frame[i])

{

pos[i]=j;

break;

}

}

}

int max1=-1;

int returnpos=-1;

for(i=0;i<nf;i++)

{

if(pos[i]>max1)

{

max1=pos[i];

returnpos=i;

}

}

return returnpos;

}

int optimal(int n, int nf, int table\_frame[],int pages[])

{

int count1=0;

cout<<"position of frame table after each request : \n";

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

{

cout<<"page table after request from "<<pages[i]<<" || ";

if(!present(table\_frame,nf,pages[i]))

{

int pos = findpos\_optimal(table\_frame,nf,pages,i,n);

table\_frame[pos]=pages[i];

printtable(table\_frame,nf);

cout<<"page fault\n";

count1++;

continue;

}

printtable(table\_frame,nf);

cout<<"\n";

}

return count1;

}

int main()

{

//nf-number of frames

int ch=0,a=1,faults=0;

while(a)

{

cout<<"\nSelect One Of The Following Page Replacement Algorithm: \n";

cout<<"\n 1. FIFO\n 2. LRU\n 3. Optimal Page Replacement Algo\n";

cout<<"\nEnter your choice : ";

cin>>ch;

int n,nf;

cout<<"Enter number of frames : ";

cin>>nf;

int table\_frame[nf];

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

{

table\_frame[i]=-1;

}

cout<<"Enter total number of page requests : ";

cin>>n;

int pages[n];

cout<<"Enter reference string : \n";

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

{

cin>>pages[i];

}

switch(ch)

{

case 1:"\nFIFO";

faults=FIFO(n,nf,table\_frame,pages);

break;

case 2:"\nLRU";

faults=LRU(n,nf,table\_frame,pages);

break;

case 3:"\nOptimal Page Replacement";

faults=optimal(n,nf,table\_frame,pages);

break;

default:"\nWrong choice.";

}

cout<<"\nTotal Number of Page Faults : "<<faults<<endl;

cout<<"\nTotal Number of Page Hit : "<<(n-faults)<<endl;

cout<<"\nDo you want to continue(1/0) : ";

cin>>a;

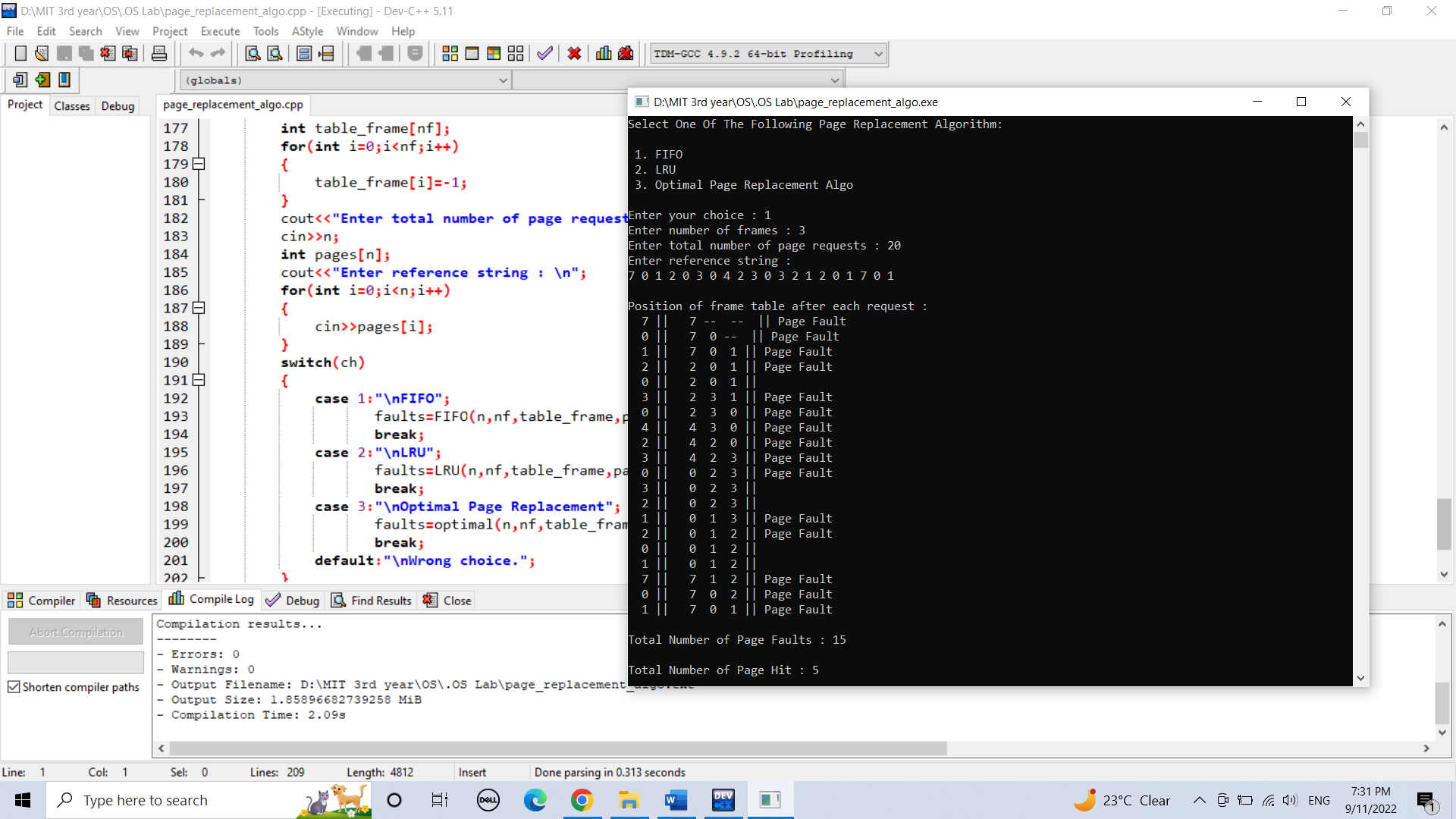
}

return 0;

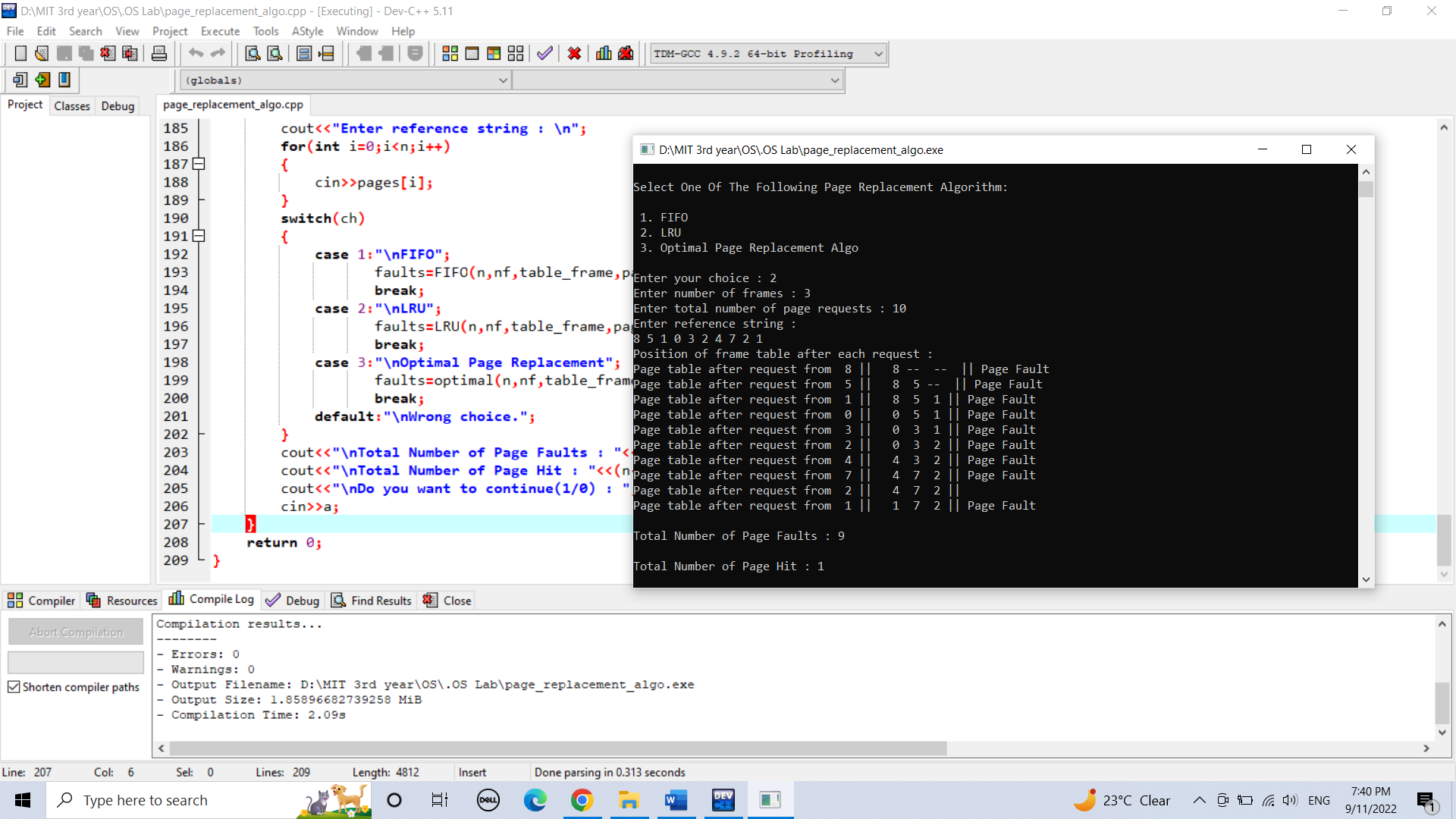
}

***Output Screenshots***

**FIFO**



**LRU**



**Optimal Page Replacement**

