***Page Replacement Algorithm***

***First In First Out Algorithm***

///Page replacement First in First Out Alogorithm

///Adnan Ismail Shah Muzavor

#include<stdio.h>

#include<stdlib.h>

int Check\_if\_in(int n,int \*frame,int key)

{

int i;

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

{if(frame[i]==key) return 1;

return 0;

}

void display(int n,int pageno,int \*frame)

{

int i;

printf(" %d | ",pageno);

for(i=0; i<n; i++) frame[i]<0?printf("%d ",frame[i]):printf(" %d ",frame[i]);

printf("\n------------------------\n");

}

void FIFO(int n,int fsz,int \*frame,int \*pgs)

{

printf("\n\_\_Applying Page Replacement First Come First Serve Algorithm\_\_");

printf("\n------------------------\n");

printf(" Page | Frame");

printf("\n------------------------");

int inserter=0,i,faults=0;

printf("\n");

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

{

if(!Check\_if\_in(fsz,frame,pgs[i]))

{

frame[inserter]=pgs[i];

inserter=(inserter+1)%fsz;

faults++;

}

display(fsz,pgs[i],frame);

}

printf("\n------------------------\nNumber of page Faults: %d\n------------------------",faults);

}

int main()

{

int n,i,fsz;

printf("Enter the number of pages referenced: ");

scanf("%d",&n);

int \*pgs=(int\*)malloc(sizeof(int)\*n);

printf("\nEnter the pages that are referenced: ");

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

{

scanf("%d",&pgs[i]);

}

printf("\nEnter the number of frames: ");

scanf("%d",&fsz);

int \*frame=(int\*)malloc(sizeof(int)\*fsz);

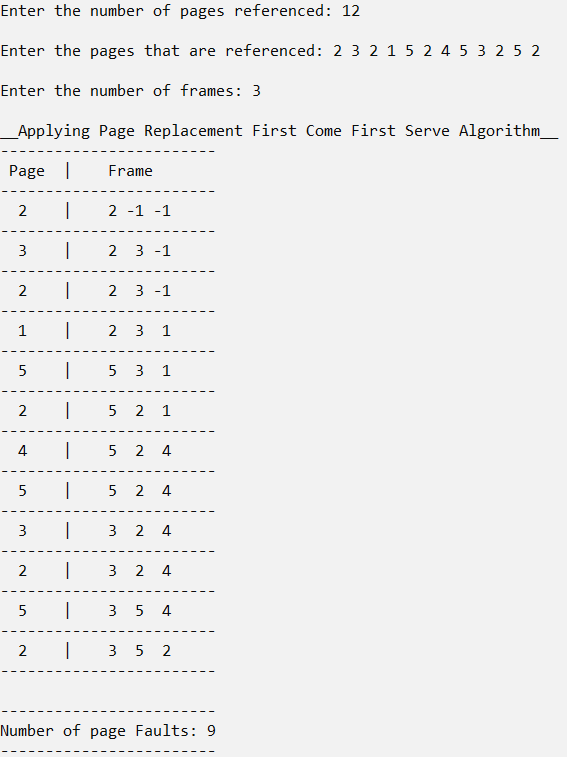
for(i=0; i<fsz; i++) frame[i]=-1;

FIFO(n,fsz,frame,pgs);

return 1;

}

***Output:***



***Least Recently Used Algorithm***

///Page replacement Least Recently Used Alogorithm

///Adnan Ismail Shah Muzavor

#include<stdio.h>

#include<stdlib.h>

#define MAX 10000

int Check\_if\_in(int n,int \*frame,int key)

{

int i;

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

{if(frame[i]==key) return 1; }

return 0;

}

int Give\_index(int key,int start,int \*pages)

{

int i;

for(i=start; i>=0; i--)

{if(pages[i]==key) return i; }

return MAX;

}

void display(int n,int pageno,int \*frame)

{

int i;

printf(" %d | ",pageno);

for(i=0; i<n; i++) frame[i]<0?printf("%d ",frame[i]):printf(" %d ",frame[i]);

printf("\n------------------------\n");

}

void LRU(int n,int fsz,int \*frame,int \*pgs)

{

printf("\n\_\_Applying Page Replacement Least Recently Used Algorithm\_\_");

printf("\n------------------------\n");

printf(" Page | Frame");

printf("\n------------------------");

int inserter=0,i,faults=0,j;

int lrf\_index=MAX,index\_to\_insert=-1;

printf("\n");

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

{

if(!Check\_if\_in(fsz,frame,pgs[i]))

{

int lrf\_index=MAX,index\_to\_insert=-1,flag\_direct=0;

for(j=0; j<fsz; j++)

{

///If any place in frame is empty, no need to do further procedure

if(frame[j]==-1)

{

frame[j]=pgs[i];

flag\_direct=1;

break;

}

///Find least occurence of this elemnt from current index

int index=Give\_index(frame[j],i,pgs);

///If these element was least referneced the prev elemnt we found

///then update the index\_to\_insert index with index of current least refrence element i.e j

if(index<lrf\_index)

{

lrf\_index=index;

index\_to\_insert=j;

}

}

///If no direct insertion was done

if(!flag\_direct)

{

///If non of the elemnts were least referenced, means insert at any position

if(index\_to\_insert==-1)

{frame[0]=pgs[i]; }

///Insert at position of least referenced element

else

{frame[index\_to\_insert]=pgs[i]; }

}

faults++;

}

display(fsz,pgs[i],frame);

}

printf("\n------------------------\nNumber of page Faults: %d\n------------------------",faults);

}

int main()

{

int n,i,fsz;

printf("Enter the number of pages referenced: ");

scanf("%d",&n);

int \*pgs=(int\*)malloc(sizeof(int)\*n);

printf("\nEnter the pages that are referenced: ");

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

{

scanf("%d",&pgs[i]);

}

printf("\nEnter the number of frames: ");

scanf("%d",&fsz);

int \*frame=(int\*)malloc(sizeof(int)\*fsz);

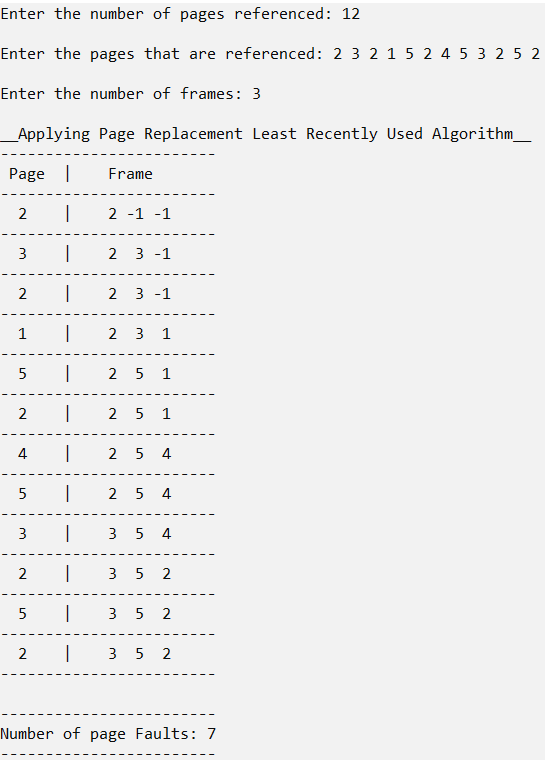
for(i=0; i<fsz; i++) frame[i]=-1;

LRU(n,fsz,frame,pgs);

return 1;

}

***Output:***

******

***Optimal Algorithm***

///Page replacement Optimal Alogorithm

///Adnan Ismail Shah Muzavor

#include<stdio.h>

#include<stdlib.h>

#define MIN -10000

#define MAX 10000

int Check\_if\_in(int n,int \*frame,int key)

{

int i;

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

{if(frame[i]==key) return 1; }

return 0;

}

int Give\_index(int key,int start,int n,int \*pages)

{

int i;

for(i=start; i<n; i++)

{if(pages[i]==key) return i; }

return MAX;

}

void display(int n,int pageno,int \*frame)

{

int i;

printf(" %d | ",pageno);

for(i=0; i<n; i++) frame[i]<0?printf("%d ",frame[i]):printf(" %d ",frame[i]);

printf("\n------------------------\n");

}

void LRU(int n,int fsz,int \*frame,int \*pgs)

{

printf("\n\_\_Applying Page Replacement Optimal Algorithm\_\_");

printf("\n------------------------\n");

printf(" Page | Frame");

printf("\n------------------------");

int inserter=0,i,faults=0,j;

int lrf\_index=MIN,index\_to\_insert=-1;

printf("\n");

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

{

if(!Check\_if\_in(fsz,frame,pgs[i]))

{

int lrf\_index=MIN,index\_to\_insert=-1,flag\_direct=0;

for(j=0; j<fsz; j++)

{

///If any place in frame is empty, no need to do further procedure

if(frame[j]==-1)

{

frame[j]=pgs[i];

flag\_direct=1;

break;

}

///Find least occurence of this elemnt from current index

int index=Give\_index(frame[j],i,n,pgs);

///If these element was least referneced the prev elemnt we found

///then update the index\_to\_insert index with index of current least refrence element i.e j

if(index>lrf\_index)

{

lrf\_index=index;

index\_to\_insert=j;

}

}

///If no direct insertion was done

if(!flag\_direct)

{

///If non of the elemnts were least referenced, means insert at any position

if(index\_to\_insert==-1)

{frame[0]=pgs[i]; }

///Insert at position of least referenced element

else

{frame[index\_to\_insert]=pgs[i]; }

}

faults++;

}

display(fsz,pgs[i],frame);

}

printf("\n------------------------\nNumber of page Faults: %d\n------------------------",faults);

}

int main()

{

int n,i,fsz;

printf("Enter the number of pages referenced: ");

scanf("%d",&n);

int \*pgs=(int\*)malloc(sizeof(int)\*n);

printf("\nEnter the pages that are referenced: ");

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

{

scanf("%d",&pgs[i]);

}

printf("\nEnter the number of frames: ");

scanf("%d",&fsz);

int \*frame=(int\*)malloc(sizeof(int)\*fsz);

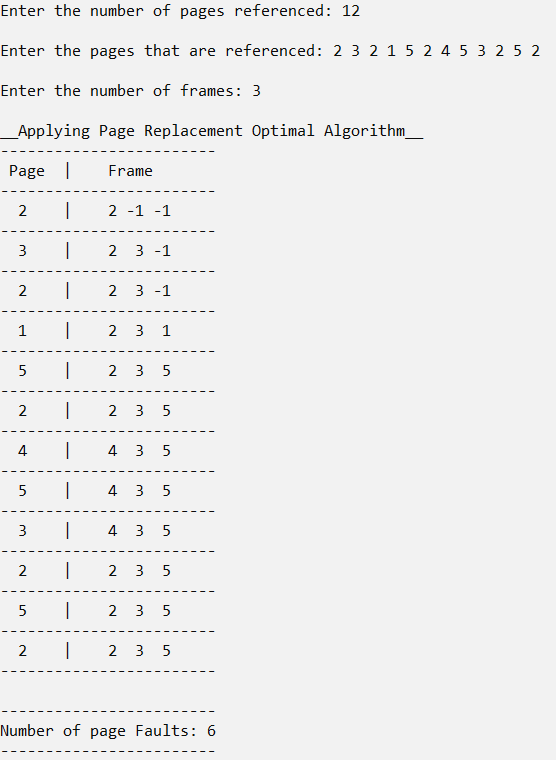
for(i=0; i<fsz; i++) frame[i]=-1;

LRU(n,fsz,frame,pgs);

return 1;

}

***Output:***

******