LAB-6

1. MFT

```
#include<iostream>
#include<stdlib.h>
#include<string.h>

#define MAX_SIZE 100
using namespace std;
struct mem{
        int* p;
        int size;
        bool p_status;
};
int main(){
        int n,occ_space=0,i=0,j=1,p_alloc;
        cout<<"\nenter the no. of processes you want ";
        cin>>n;
        int int_frag[n],ext_frag=0;
        struct mem* arr=new mem[n];

        for(int x=0;x<n;x++)
                int_frag[x]=0;

        for(int x=0;x<n;x++)
                arr[x].p_status=false;

        while(i!=n){
                if(occ_space<=MAX_SIZE && i<(n-1)){
                        cout<<"\nenter initial mem_space of process: ";
                        cin>>p_alloc;
                        arr[i].size=p_alloc;
                        occ_space=occ_space+arr[i].size;
                        cout<<"remaining memory is: "<<(MAX_SIZE-occ_spac
e);
                        i++;
                }
                else if(occ_space<=MAX_SIZE && i==(n-1)){
                        cout<<"\nautomatic remaining mem_space alloacted
to last process. ";
```

```
                        arr[i].size=(MAX_SIZE-occ_space);
                        occ_space=occ_space+arr[i].size;
                        cout<<"\nremaining memory is: "<<(MAX_SIZE-occ_sp
ace);
                        i++;
                }
                else
                        occ_space=occ_space-arr[i].size;
        }
        cout<<"\nmemory allocated to different processes are: ";
        for(int k=0;k<n;k++)
                cout<<arr[k].size<<"  ";

        cout<<"\n\ninitial mem_space for diff. processes have been alloca
ted,now proceed... \n";

        for(int x=0;x<n;x++)
                arr[x].p=new int[arr[x].size];

        for(int x=0;x<n;x++)
                for(int y=0;y<arr[x].size;y++)
                        arr[x].p[y]=0;

        while(j!=(n+1)){
                bool status=false;
                int p_size;
                cout<<"\nEnter process size: ";
                cin>>p_size;
                for(int x=0 ; x<n && status==false ; x++){
                        if(arr[x].size>=p_size && arr[x].p_status==false)
{
                                for(int y=0;y<p_size;y++){
                                        arr[x].p[y]=j;
                                }
                                arr[x].p_status=true;
```

```
                                j++;
                                status=true;
                        }
                }
                if(status==false){
                        cout<<"\np_size is large,please enter another p_s
ize value: ";
                }
        }
        cout<<"\nmem_allocation list is: \n";
        for(int x=0;x<n;x++){
                for(int y=0;y<arr[x].size;y++){
                        cout<<arr[x].p[y]<<" ";
                }
                cout<<"\n";
        }

        for(int x=0;x<n;x++){
                for(int y=0;y<arr[x].size;y++){
                        if(arr[x].p[y]==0)
                                int_frag[x]=int_frag[x]+1;
                }
        }

        cout<<"\nInternal fragmentation is: ";
        for(int x=0;x<n;x++)
                cout<<int_frag[x]<<"  ";

        cout<<"\nExternal fragmentation is: ";
        for(int x=0;x<n;x++)
                ext_frag=ext_frag+int_frag[x];
        cout<<ext_frag<<"\n\n";

        return 0;
}
```

```
<131352@Linux-Svr ~/5sem/lab6>$./a.out

enter the no. of processes you want 5

enter initial mem_space of process: 25
remaining memory is: 75
enter initial mem_space of process: 35
remaining memory is: 40
enter initial mem_space of process: 15
remaining memory is: 25
enter initial mem_space of process: 5
remaining memory is: 20
automatic remaining mem_space alloacted to last process.
remaining memory is: 0
memory allocated to different processes are: 25  35  15  5  20

initial mem_space for diff. processes have been allocated,now proceed...

Enter process size: 17

Enter process size: 23

Enter process size: 9

Enter process size: 7

Enter process size: 13

p_size is large,please enter another p_size value:
Enter process size: 4
```

```
mem_allocation list is:
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
3 3 3 3 3 3 3 3 3 0 0 0 0 0 0
5 5 5 5 0
4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Internal fragmentation is: 8  12  6  1  13
External fragmentation is: 40
```

2. Fifo_MVT

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

int main(){
        int MAX_SIZE;

        vector<int>int_frag;
        int  int_frag_count=0,sum=0;
        vector<int>v;
        vector<int>fchunk;

        cout<<"\nEnter the total mem_size for allocation: ";
        cin>>MAX_SIZE;

        int* arr=new int[MAX_SIZE];
        for(int i=0;i<MAX_SIZE;i++)
                arr[i]=0;

        int size,z,ext_frag=0,v_loc;
        int p_num=1,fchunk_count=0;;
        char ch='y',ch2='n';

        while(ch!='n'){
                int count=0;
                bool status=false;
                bool mem_status=false;
                cout<<"\nenter the process "<<p_num<<" size: ";

                v.push_back(p_num);
                cin>>size;
```

```
                for(int i=0 ; i<MAX_SIZE ; i++){
                        if(arr[i]==0){
                                int temp=i;
                                count++;
                                if(count==size && mem_status!=true){
                                        for(int i=temp+1-size ; i<(temp+1
-size+count) ; i++){
                                                arr[i]=p_num;
                                                mem_status=true;
                                        }
                                }
                        }
                        else{
                                count=0;
                        }
                }
                if(mem_status!=true){
                        cout<<"\nNo memory availble...Need to terminate s
ome process...";
                        cout<<"\nDo you want to terminate any process('y'
/'n'): ";
                        cin>>ch2;
                        bool v_status=false;

                        if(ch2=='y'){
                                cout<<"\nwhich process do you want to ter
minate:(int) ";
                                cin>>z;

                                for(int x=0 ; x<v.size() && v_status==fal
se ; x++){
                                        if(v[x]==z){
                                                v_status=true;
                                                v_loc=x;
                                        }
                                }
```

```
                                        if(v_status==true){
                                                cout<<"\nTerminate process "<<v[v
_loc];
                                                for(int i=0 ; i<MAX_SIZE && statu
s!=true ; i++){
                                                        if(arr[i]==v[v_loc])
                                                                status=true;
                                                        else
                                                                status=false;
                                                }
                                                if(status==true){

                                                        for(int i=0;i<MAX_SIZE;i+
+){
                                                                if(arr[i]==v[v_lo
c])
                                                                        arr[i]=0;
                                                        }
                                                }
                                        }
                                        else
                                                cout<<"\nprocess"<<z<<" doesn't e
xist in memory";
                                }
                        }
                        else{
                                p_num++;
                        }

                        cout<<"\n";

                        for(int i=0;i<MAX_SIZE;i++)
                                cout<<arr[i]<<" ";
```

```
                cout<<"\nDo you want to continue('y'/'n'): ";
                cin>>ch;
        }
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0){
                        int_frag_count++;
                        if(int_frag_count==2 || int_frag_count==1){
                                if(i<(MAX_SIZE-1) && arr[i+1]!=0)
                                        int_frag.push_back(int_frag_count
);
                                else if(i==(MAX_SIZE-1) && arr[i]==0)
                                        int_frag.push_back(int_frag_count
);
                        }
                }
                else
                        int_frag_count=0;
        }

        cout<<"\nFree chunk list: ";
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0){
                        fchunk_count++;
                        if(i<(MAX_SIZE-1) && arr[i+1]!=0)
                                fchunk.push_back(fchunk_count);
                        else if(i==MAX_SIZE-1)
                                fchunk.push_back(fchunk_count);
                }
                else
                        fchunk_count=0;
        }
        for(int i=0;i<fchunk.size();i++)
                cout<<fchunk[i]<<"  ";

        cout<<"\n";
```

```
        cout<<"\nTotal internal fragmentation is: ";
        for(int i=0;i<int_frag.size();i++)
                sum=sum+int_frag[i];
        cout<<sum;

        cout<<"\nexternal fragmentation is: ";
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0)
                        ext_frag=ext_frag+1;
        }
        cout<<(ext_frag-sum)<<"\n\n";

        return 0;
}
```

```
<131352@Linux-Svr ~/5sem/lab6>$g++ fifo_MVT.cpp
<131352@Linux-Svr ~/5sem/lab6>$./a.out

Enter the total mem_size for allocation: 20

enter the process 1 size: 5

1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 2 size: 7

1 1 1 1 1 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 3 size: 3

1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 4 size: 4

1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 4 4 4 4 0
Do you want to continue('y'/'n'): y

enter the process 5 size: 7

No memory availble...Need to terminate some process...
Do you want to terminate any process('y'/'n'): y

which process do you want to terminate:(int) 2

Terminate process 2
1 1 1 1 1 0 0 0 0 0 0 0 3 3 3 4 4 4 4 0
Do you want to continue('y'/'n'): y
```

```
enter the process 5 size: 4

1 1 1 1 1 5 5 5 5 0 0 0 3 3 3 4 4 4 4 0
Do you want to continue('y'/'n'): y

enter the process 6 size: 7

No memory availble...Need to terminate some process...
Do you want to terminate any process('y'/'n'): y

which process do you want to terminate:(int) 1

Terminate process 1
0 0 0 0 0 5 5 5 5 0 0 0 3 3 3 4 4 4 4 0
Do you want to continue('y'/'n'): y

enter the process 6 size: 3

6 6 6 0 0 5 5 5 5 0 0 0 3 3 3 4 4 4 4 0
Do you want to continue('y'/'n'): n

Free chunk list: 2  3  1

Total internal fragmentation is: 3
external fragmentation is: 3
```

3. best_fit_MVT

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<utility>
using namespace std;

bool sortbyfirst(const pair<int,int> &a,const pair<int,int> &b){
        return (a.first < b.first);
}

int main(){
        int MAX_SIZE;

        vector<int>int_frag;
        int  int_frag_count=0,sum=0;
        vector<int>v;
        vector<int>fchunk;
        vector<pair<int,int> > tars;

        cout<<"\nEnter the total mem_size for allocation: ";
        cin>>MAX_SIZE;

        int* arr=new int[MAX_SIZE];
        for(int i=0;i<MAX_SIZE;i++)
                arr[i]=0;

        int size,z,ext_frag=0,v_loc;
        int p_num=1,fchunk_count=0;;
        char ch='y',ch2='n';

        while(ch!='n'){
                int count=0,temp=0;
                bool status=false;
                bool mem_status=false;
                cout<<"\nenter the process "<<p_num<<" size: ";
```

```
                v.push_back(p_num);
                cin>>size;

                for(int i=0 ; i<MAX_SIZE ; i++){
                        if(arr[i]==0){
                                temp=i;
                                count++;
                                if(count>2 && i<MAX_SIZE-1 && arr[i+1]!=0
){
                                        tars.push_back(make_pair(count,te
mp));
                                }
                                else if(count>2 && i==MAX_SIZE-1){
                                        tars.push_back(make_pair(count,te
mp));
                                }
                        }
                        else{
                                count=0;
                        }
                }

                stable_sort(tars.begin(),tars.end(),sortbyfirst);
```

```
                for(vector<pair<int,int> >::iterator it=tars.begin() ; it
!=tars.end() && mem_status!=true ; it++){
                        if(it->first >= size){
                                for(int i=0;i<size;i++){
                                        arr[it->second + 1 - it->first +
i]=p_num;
                                }
                                mem_status=true;
                        }
                }

                tars.erase(tars.begin(),tars.end());

                if(mem_status!=true){
                        cout<<"\nNo memory availble...Need to terminate s
ome process...";
                        cout<<"\nDo you want to terminate any process('y'
/'n'): ";
                        cin>>ch2;
                        bool v_status=false;

                        if(ch2=='y'){
                                cout<<"\nwhich process do you want to ter
minate:(int) ";
                                cin>>z;

                                for(int x=0 ; x<v.size() && v_status==fal
se ; x++){
                                        if(v[x]==z){
                                                v_status=true;
                                                v_loc=x;
                                        }
                                }
```

```
                                if(v_status==true){
                                        cout<<"\nTerminate process"<<v[v_
loc];
                                        for(int i=0 ; i<MAX_SIZE && statu
s!=true ; i++){
                                                if(arr[i]==v[v_loc])
                                                        status=true;
                                                else
                                                        status=false;
                                        }
                                        if(status==true){
                                                for(int i=0;i<MAX_SIZE;i+
+){
                                                        if(arr[i]==v[v_lo
c])
                                                                arr[i]=0;
                                                }
                                        }
                                }
                                else
                                        cout<<"\nprocess"<<z<<" doesn't e
xist in memory";
                        }
                }
                else{
                        p_num++;
                }

                cout<<"\n";

                for(int i=0;i<MAX_SIZE;i++)
                        cout<<arr[i]<<" ";

                cout<<"\nDo you want to continue('y'/'n'): ";
                cin>>ch;
        }
```

```
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0){
                        int_frag_count++;
                        if(int_frag_count==2 || int_frag_count==1){
                                if(i<(MAX_SIZE-1) && arr[i+1]!=0)
                                        int_frag.push_back(int_frag_count
);
                                else if(i==(MAX_SIZE-1) && arr[i]==0)
                                        int_frag.push_back(int_frag_count
);
                        }
                }
                else
                        int_frag_count=0;
        }

        cout<<"\nFree chunk list: ";
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0){
                        fchunk_count++;
                        if(i<(MAX_SIZE-1) && arr[i+1]!=0)
                                fchunk.push_back(fchunk_count);
                        else if(i==MAX_SIZE-1)
                                fchunk.push_back(fchunk_count);
                }
                else
                        fchunk_count=0;
        }
        for(int i=0;i<fchunk.size();i++)
                cout<<fchunk[i]<<"  ";

        cout<<"\n";
```

```
        cout<<"\nTotal internal fragmentation is: ";
        for(int i=0;i<int_frag.size();i++)
                sum=sum+int_frag[i];
        cout<<sum;

        cout<<"\nexternal fragmentation is: ";
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0)
                        ext_frag=ext_frag+1;
        }
        cout<<(ext_frag-sum)<<"\n\n";

        return 0;
}
```

```
<131352@Linux-Svr ~/5sem/lab6>$g++ bestfit_MVT.cpp
<131352@Linux-Svr ~/5sem/lab6>$./a.out

Enter the total mem_size for allocation: 20

enter the process 1 size: 5

1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 2 size: 7

1 1 1 1 1 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 3 size: 6

1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 0 0
Do you want to continue('y'/'n'): y

enter the process 4 size: 7

No memory availble...Need to terminate some process...
Do you want to terminate any process('y'/'n'): y

which process do you want to terminate:(int) 2

Terminate process2
1 1 1 1 1 0 0 0 0 0 0 0 3 3 3 3 3 3 0 0
Do you want to continue('y'/'n'): y

enter the process 4 size: 3
```

```
1 1 1 1 1 4 4 4 0 0 0 0 3 3 3 3 3 3 0 0
Do you want to continue('y'/'n'): y

enter the process 5 size: 11

No memory availble...Need to terminate some process...
Do you want to terminate any process('y'/'n'): y

which process do you want to terminate:(int) 1

Terminate process1
0 0 0 0 0 4 4 4 0 0 0 0 3 3 3 3 3 3 0 0
Do you want to continue('y'/'n'): y

enter the process 5 size: 3

0 0 0 0 0 4 4 4 5 5 5 0 3 3 3 3 3 3 0 0
Do you want to continue('y'/'n'): n

Free chunk list: 5  1  2

Total internal fragmentation is: 3
external fragmentation is: 5
```

4. worst_fit_MVT

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<utility>
using namespace std;

bool sortbyfirst(const pair<int,int> &a,const pair<int,int> &b){
        return (a.first > b.first);
}

int main(){
        int MAX_SIZE;

        vector<int>int_frag;
        int  int_frag_count=0,sum=0;
        vector<int>v;
        vector<int>fchunk;
        vector<pair<int,int> > tars;

        cout<<"\nEnter the total mem_size for allocation: ";
        cin>>MAX_SIZE;

        int* arr=new int[MAX_SIZE];
        for(int i=0;i<MAX_SIZE;i++)
                arr[i]=0;

        int size,z,ext_frag=0,v_loc;
        int p_num=1,fchunk_count=0;;
        char ch='y',ch2='n';

        while(ch!='n'){
                int count=0,temp=0;
                bool status=false;
                bool mem_status=false;
                cout<<"\nenter the process "<<p_num<<" size: ";
```

```
                v.push_back(p_num);
                cin>>size;

                for(int i=0 ; i<MAX_SIZE ; i++){
                        if(arr[i]==0){
                                temp=i;
                                count++;
                                if(count>2 && i<MAX_SIZE-1 && arr[i+1]!=0
){
                                        tars.push_back(make_pair(count,te
mp));
                                }
                                else if(count>2 && i==MAX_SIZE-1){
                                        tars.push_back(make_pair(count,te
mp));
                                }
                        }
                        else{
                                count=0;
                        }
                }

                stable_sort(tars.begin(),tars.end(),sortbyfirst);

                for(vector<pair<int,int> >::iterator it=tars.begin() ; it
!=tars.end() && mem_status!=true ; it++){
                        if(it->first >= size){
                                for(int i=0;i<size;i++){
                                        arr[it->second + 1 - it->first +
i]=p_num;
                                }
                                mem_status=true;
                        }
                }
```

```
                tars.erase(tars.begin(),tars.end());

                if(mem_status!=true){
                        cout<<"\nNo memory availble...Need to terminate s
ome process...";
                        cout<<"\nDo you want to terminate any process('y'
/'n'): ";
                        cin>>ch2;
                        bool v_status=false;

                        if(ch2=='y'){
                                cout<<"\nwhich process do you want to ter
minate:(int) ";
                                cin>>z;

                                for(int x=0 ; x<v.size() && v_status==fal
se ; x++){
                                        if(v[x]==z){
                                                v_status=true;
                                                v_loc=x;
                                        }
                                }

                                if(v_status==true){
                                        cout<<"\nTerminate process"<<v[v_
loc];
                                        for(int i=0 ; i<MAX_SIZE && statu
s!=true ; i++){
                                                if(arr[i]==v[v_loc])
                                                        status=true;
                                                else
                                                        status=false;
                                        }
                                        if(status==true){
                                                for(int i=0;i<MAX_SIZE;i+
+){
                                                        if(arr[i]==v[v_lo
c])
                                                                arr[i]=0;
                                                }
                                        }
                                }
                                else
                                        cout<<"\nprocess"<<z<<" doesn't e
xist in memory";
                        }
                }
                else{
                        p_num++;
                }

                cout<<"\n";

                for(int i=0;i<MAX_SIZE;i++)
                        cout<<arr[i]<<" ";
```

```
                cout<<"\nDo you want to continue('y'/'n'): ";
                cin>>ch;
        }
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0){
                        int_frag_count++;
                        if(int_frag_count==2 || int_frag_count==1){
                                if(i<(MAX_SIZE-1) && arr[i+1]!=0)
                                        int_frag.push_back(int_frag_count
);
                                else if(i==(MAX_SIZE-1) && arr[i]==0)
                                        int_frag.push_back(int_frag_count
);
                        }
                }
                else
                        int_frag_count=0;
        }

        cout<<"\nFree chunk list: ";
        for(int i=0;i<MAX_SIZE;i++){
                if(arr[i]==0){
                        fchunk_count++;
                        if(i<(MAX_SIZE-1) && arr[i+1]!=0)
                                fchunk.push_back(fchunk_count);
                        else if(i==MAX_SIZE-1)
                                fchunk.push_back(fchunk_count);
                }
                else
                        fchunk_count=0;
        }
        for(int i=0;i<fchunk.size();i++)
                cout<<fchunk[i]<<"  ";
```

```
          cout<<"\n";

          cout<<"\nTotal internal fragmentation is: ";
          for(int i=0;i<int_frag.size();i++)
                  sum=sum+int_frag[i];
          cout<<sum;

          cout<<"\nexternal fragmentation is: ";
          for(int i=0;i<MAX_SIZE;i++){
                  if(arr[i]==0)
                          ext_frag=ext_frag+1;
          }
          cout<<(ext_frag-sum)<<"\n\n";

          return 0;
}
```

```
<131352@Linux-Svr ~/5sem/lab6>$g++ worstfit_MVT.cpp
<131352@Linux-Svr ~/5sem/lab6>$./a.out

Enter the total mem_size for allocation: 20

enter the process 1 size: 5

1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 2 size: 7

1 1 1 1 1 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 3 size: 6

1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 0 0
Do you want to continue('y'/'n'): y

enter the process 4 size: 7

No memory availble...Need to terminate some process...
Do you want to terminate any process('y'/'n'): y

which process do you want to terminate:(int) 3

Terminate process3
1 1 1 1 1 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 4 size: 11
```

```
No memory availble...Need to terminate some process...
Do you want to terminate any process('y'/'n'): y

which process do you want to terminate:(int) 1

Terminate process1
0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 4 size: 3

0 0 0 0 0 2 2 2 2 2 2 2 4 4 4 0 0 0 0 0
Do you want to continue('y'/'n'): y

enter the process 5 size: 4

5 5 5 5 0 2 2 2 2 2 2 2 4 4 4 0 0 0 0 0
Do you want to continue('y'/'n'): n

Free chunk list: 1  5

Total internal fragmentation is: 1
external fragmentation is: 5
```

LAB – 7

1. FIFO

```
#include<iostream>
#include<limits.h>
using namespace std;
struct queue
{
        int front,rear,size,capacity;
        int* array;
};
struct queue* create(int cap)
{
        struct queue* q=new queue;
        q->capacity=cap;
        q->front=q->size=0;
        q->rear=-1;
        q->array=new int[q->capacity];

        for(int i=0;i<q->capacity;i++)
                q->array[i]=-1;

        return q;
}
bool isfull(struct queue* q)
{
        if(q->size==q->capacity)
                return true;
        else
                return false;
}
bool isempty(struct queue* q)
{
        if(q->size==0)
                return true;
        else
                return false;
}
void enq(struct queue* q,int item)
{
```

```
        if(isfull(q))
                return;
        q->rear=(q->rear+1)%q->capacity;
        q->array[q->rear]=item;
        q->size=q->size+1;
}
int deq(struct queue* q)
{
        if(isempty(q))
                return INT_MIN;
        int item=q->array[q->front];
        q->front=(q->front+1)%q->capacity;
        q->size=q->size-1;
        return item;
}
bool find(int* arr,int n,int x)
{
        for(int i=0;i<n;i++)
        {
                if(arr[i]==x)
                        return true;
                else
                        continue;
        }
        return false;
}

int main()
{
        int page_fault=0;
        struct queue* q=create(3);
        int str[]={7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1};
        int x=sizeof(str)/sizeof(str[0]);
```

```
        for(int i=0;i<x;i++)
        {
                bool state=find(q->array,q->capacity,str[i]);
                if(state==true){
                        cout<<".\n\n";
                        continue;
                }
                else
                {
                        if(isfull(q))
                        {
                                deq(q);
                                enq(q,str[i]);
                                for(int j=0;j<q->size;j++)
                                        cout<<q->array[j]<<"  ";
                                page_fault++;
                        }
                        else
                        {
                                enq(q,str[i]);
                                for(int j=0;j<q->size;j++)
                                        cout<<q->array[j]<<"  ";
                                page_fault++;
                        }
                }
                cout<<"\n";
        }
        cout<<"\nTotal no. of page fault is: "<<page_fault<<"\n";
}
```

```
<131352@Linux-Svr ~/5sem/lab7>$g++ fifo.cpp
<131352@Linux-Svr ~/5sem/lab7>$./a.out

Reference String is: 7 0 1 2 0 3 0 4 2 3 0 3 0 3 2 1 2 0 1 7 0 1

7
7  0
7  0  1
2  0  1
.

2  3  1
2  3  0
4  3  0
4  2  0
4  2  3
0  2  3
.

.

.

.

0  1  3
0  1  2
.

.

7  1  2
7  0  2
7  0  1

Total no. of page fault is: 15
```

2. Optimal

```cpp
#include<iostream>
#include<vector>
#include<algorithm>
#include<utility>
#include<limits.h>
using namespace std;

typedef vector<pair<int,int> >::iterator vit;

bool foo_neg(const pair<int,int> &a)
{
        return ((a.second < 0)==true);
}

bool foo2(const pair<int,int> &a,const pair<int,int> &b)
{
        return ((a.second < b.second)==true);
}

bool v_search(vector<pair<int,int> > v3,int item)
{
        for(vit it=v3.begin() ; it!=v3.end() ; it++)
        {
                if(it->first==item)
                        return true;
                else
                        continue;
        }
        return false;
}
```

```
vit v_search2(vector<pair<int,int> > &v5,int item)
{
        for(vit it=v5.begin() ; it!=v5.end() ; it++)
        {
                if(it->first==item)
                        return it;
        }
}

int main(){
        int page_fault=0;

        vector<pair<int,int> > v(3);
        for(int i=0;i<v.size();i++)
                v[i]=make_pair(-1,-1);

        int str[]={7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1};
        int x=sizeof(str)/sizeof(str[0]);
        vector<int> tars(str,str+x);

        cout<<"\nReference String is: ";
        for(int i=0;i<tars.size();i++)
                cout<<tars[i]<<" ";
        cout<<"\n\n";
```

```
        for(int i=0;i<tars.size();i++)
        {
                if(v_search(v,tars[i]))
                {
                        if(i<tars.size()-1 && find(tars.begin()+i+1,tars.
end(),tars[i])!=tars.end())
                        {
                                int index=find(tars.begin()+i+1,tars.end(
),tars[i])-tars.begin();
                                vit it = v_search2(v,tars[i]);          //
here we use v_search2();
                                it->second=index;
                                cout<<".\n";
                                continue;
                        }
                        else
                        {
                                vit it = v_search2(v,tars[i]);          //
here we use v_search2();
                                it->second=INT_MAX;
                                cout<<".\n";
                                continue;
                        }
                }
                else
                {
                        if(i<tars.size()-1 && find(tars.begin()+i+1,tars.
end(),tars[i])!=tars.end())
                        {
                                int index=find(tars.begin()+i+1,tars.end(
),tars[i])-tars.begin();

                                if(find_if(v.begin(),v.end(),foo_neg)!=v.
end())     //this condition is only for initial three vector v fill-up
                                {
                                        vit it = find_if (v.begin(), v.en
d(), foo_neg);
```

```
                                        *it=make_pair(tars[i],index);
                                        page_fault++;

                                        for(int j=0;j<v.size() && v[j].fi
rst!=-1;j++)
                                                cout<<v[j].first<<"  ";
                                }
                                else{
                                        vit it = max_element (v.begin(),
v.end(), foo2);
                                        *it=make_pair(tars[i],index);
                                        page_fault++;

                                        for(int j=0;j<v.size() && v[j].fi
rst!=-1;j++)
                                                cout<<v[j].first<<"  ";
                                }
                        }
                        else
                        {
                                if(find_if(v.begin(),v.end(),foo_neg)!=v.
end())    //this condition is only for initial three vector v fill-up
                                {
                                        vit it = find_if (v.begin(), v.en
d(), foo_neg);
                                        *it=make_pair(tars[i],INT_MAX);
                                        page_fault++;

                                        for(int j=0;j<v.size() && v[j].fi
rst!=-1;j++)
                                                cout<<v[j].first<<"  ";
                                }
                                else
                                {
                                        vit it = max_element (v.begin(),
v.end(), foo2);
                                        *it=make_pair(tars[i],INT_MAX);
                                        page_fault++;

                                        for(int j=0;j<v.size() && v[j].fi
rst!=-1;j++)
                                                cout<<v[j].first<<"  ";
                                }
                        }
                }
                cout<<"\n";
        }
        cout<<"\nTotal page fault is: "<<page_fault<<"\n\n";
}
```

```
<131352@Linux-Svr ~/5sem/lab7>$g++ optimal.cpp
<131352@Linux-Svr ~/5sem/lab7>$./a.out

Reference String is: 7 0 1 2 0 3 0 4 2 3 0 3 0 3 2 1 2 0 1 7 0 1

7
7  0
7  0  1
2  0  1
.
2  0  3
.
2  4  3
.
.
2  0  3
.
.
.
.
2  0  1
.
.
.
7  0  1
.
.

Total page fault is: 9
```

3. LRU

```
#include<iostream>
#include<vector>
#include<list>
#include<algorithm>
#include<utility>
#include<limits.h>
using namespace std;

typedef list<int>::iterator lit;

bool v_search(list<int> &v3,int item)
{
        for(lit it=v3.begin() ; it!=v3.end() ; it++)
        {
                if(*it==item)
                        return true;
                else
                        continue;
        }
        return false;
}

lit v_search2(list<int> &v5,int item)
{
        for(lit it=v5.begin() ; it!=v5.end() ; it++)
        {
                if(*it==item)
                        return it;
        }
}

int main(){
        int page_fault=0;
```

```
        list<int> v(3);
        for(lit it=v.begin() ; it!=v.end() ; it++)
                *it=-1;

        int str[]={7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1};
        int x=sizeof(str)/sizeof(str[0]);
        vector<int> tars(str,str+x);

        cout<<"\nReference String is: ";
        for(int i=0;i<tars.size();i++)
                cout<<tars[i]<<" ";
        cout<<"\n\n";

        for(int i=0;i<tars.size();i++)
        {
                if(v_search(v,tars[i]))
                {
                        lit it=v_search2(v,tars[i]);
                        v.erase(it);
                        v.push_back(tars[i]);
                        cout<<".\n";
                        continue;
                }
                else
                {
                        if(find(v.begin(),v.end(),-1)!=v.end())
                        {
                                lit it=find(v.begin(),v.end(),-1);
                                *it=tars[i];
                                page_fault++;

                                for(lit it=v.begin() ; it!=v.end() && *it
!=-1 ; it++)
                                        cout<<*it<<"  ";
                        }
                        else
                        {
                                v.erase(v.begin());
                                v.push_back(tars[i]);
                                page_fault++;

                                for(lit it=v.begin() ; it!=v.end() ; it++
)
                                        cout<<*it<<"  ";
                        }
                }
                cout<<"\n";
        }
        cout<<"\nTotal page fault is: "<<page_fault<<"\n\n";
}
```

```
<131352@Linux-Svr ~/5sem/lab7>$g++ lru.cpp
<131352@Linux-Svr ~/5sem/lab7>$./a.out

Reference String is: 7 0 1 2 0 3 0 4 2 3 0 3 0 3 2 1 2 0 1 7 0 1

7
7   0
7   0   1
0   1   2
.
2   0   3
.
3   0   4
0   4   2
4   2   3
2   3   0
.
.
.
.
3   2   1
.
1   2   0
.
0   1   7
.
.

Total page fault is: 12
```