1)FCFS

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

int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    int bt[n], wt[n], total_wt = 0;

    cout << "Enter burst times:\n";
    for(int i = 0; i < n; i++)
        cin >> bt[i];

    wt[0] = 0; // প্রথম প্রসেসের waiting time সবসময় 0

    for(int i = 1; i < n; i++) {
        wt[i] = wt[i-1] + bt[i-1];  // waiting time হিসাব
        total_wt += wt[i];          // waiting time যোগ করছি
    }

    cout << "Average Waiting Time: " << (float)total_wt/n << endl;
}
```

2)SJF -nonprem

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

struct Process {
    int id, bt;
};

bool compare(Process a, Process b) {
    return a.bt < b.bt;  // ছোট burst time আগে আসবে
}

int main(){
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    Process p[n];

    cout << "Enter burst times:\n";
    for(int i = 0; i < n; i++){
        p[i].id = i+1;
```

```
        cin >> p[i].bt;
    }

    sort(p, p+n, compare); // burst time অনুযায়ী sort করছি

    int wt[n]={0}, total_wt=0;

    for(int i=1; i<n; i++){
        wt[i] = wt[i-1] + p[i-1].bt; // waiting time হিসাব
        total_wt += wt[i];
    }

    cout << "Average Waiting Time: " << (float)total_wt/n << endl;
}
```

3)banker

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

int main() {
    int n, m;

    // প্রসেস এবং রিসোর্সের সংখ্যা ইনপুট নেওয়া
    cout << "Enter number of processes: ";
    cin >> n;
    cout << "Enter number of resources: ";
    cin >> m;

    int allocation[n][m], max[n][m], need[n][m];
    int available[m];

    // Allocation Matrix ইনপুট নেওয়া
    cout << "\nEnter Allocation Matrix:\n";
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            cin >> allocation[i][j];
        }
    }

    // Max Need Matrix ইনপুট নেওয়া
    cout << "\nEnter Max Matrix:\n";
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            cin >> max[i][j];
        }
    }

    // Available Resources ইনপুট নেওয়া
    cout << "\nEnter Available Resources:\n";
```

```
    for(int i=0; i<m; i++){
        cin >> available[i];
    }

    // Need Matrix গণনা করা (Need = Max - Allocation)
    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }

    bool finish[n] = {0}; // কোন প্রসেস শেষ হয়েছে কিনা ট্র্যাক করার জন্য
    int safeSeq[n];       // Safe Sequence সংরক্ষণের জন্য
    int cnt = 0;          // কয়টা প্রসেস শেষ হয়েছে তা গণনার জন্য

    while(cnt < n){
        bool found = false;
        for(int i=0; i<n; i++){
            if(finish[i]==0){
                int j;
                for(j=0; j<m; j++){
                    // যদি কোনো প্রসেসের প্রয়োজনের থেকে কম রিসোর্স থাকে
                    if(need[i][j] > available[j])
                        break;
                }

                // প্রসেসটি যদি Available Resources দিয়ে চালানো যায়
                if(j == m){
                    for(int k=0; k<m; k++)
                        available[k] += allocation[i][k]; // রিসোর্স ফেরত
নেওয়া

                    safeSeq[cnt++] = i; // প্রসেসকে সেফ সিকোয়েন্সে রাখা
                    finish[i] = true;   // প্রসেস সম্পন্ন হিসেবে মার্ক করা
                    found = true;       // অন্তত একটি প্রসেস পাওয়া গেছে
                }
            }
        }
        if(!found) break; // আর কোনো প্রসেস চালানো না গেলে ডেডলক হবে
    }

    // ডেডলক আছে কিনা চেক করা
    if(cnt != n)
        cout << "\nDeadlock detected! (ডেডলক হয়েছে!)\n";
    else{
        cout << "\nSafe Sequence (সেফ সিকোয়েন্স): ";
        for(int i=0; i<n; i++)
            cout << "P" << safeSeq[i] << " ";
        cout << endl;
    }

    return 0;
```

```
}
```

4) first-best-worst

```
#include<bits/stdc++.h>
using namespace std;
void firstFit(int holes[],int n,int processSize){
    for(int i=0; i < n; i++){
        if(holes[i]>=processSize){
            cout << "First Fit: " << holes[i] << "k" << endl;
            return;
        }

    }
    cout << "Wait Please Don't have sufficient Space" << endl;
}
void bestFit(int holes[],int n,int processSize){
    sort(holes,holes+n);
    for(int i=0; i < n; i++){
        if(holes[i]>=processSize){
            cout << "Best Fit: " << holes[i] << "k"<< endl;
            return;
        }
    }
    cout << "Wait Please Don't have sufficient Space" << endl;
}
void worstFit(int holes[],int n,int processSize){
    sort(holes,holes+n,greater<int>());
    if(processSize<=holes[0]){
        cout << "Worst Fit: " << holes[0] << "k" << endl;
        return;
    }
    cout << "Wait Please Don't have sufficient Space" << endl;
}


int main(){
    int n;
    cout << "Enter the number of Holes availabe:";
    cin >> n;

    cout << "Enter the Hole Sizes : ";
    int holes[n];
    for(int i = 0; i < n; i++){
        cin >> holes[i];
    }

    int processSize;
    cout << "Enter process size: ";
    cin >> processSize;

    firstFit(holes, n, processSize);
    bestFit(holes, n, processSize);
    worstFit(holes, n, processSize);
}
```

5)page replacement-fifo

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

int main(){
    int pages, frames, hits = 0;

    // মোট পেজের সংখ্যা ইনপুট নেওয়া
    cout << "Enter number of pages: ";
    cin >> pages;

    int ref[pages];

    // পেজ রেফারেন্স স্ট্রিং ইনপুট নেওয়া
    cout << "Enter page reference string:\n";
    for(int i=0; i<pages; i++){
        cin >> ref[i];
    }

    // ফ্রেম সংখ্যা ইনপুট নেওয়া
    cout << "Enter frame size: ";
    cin >> frames;

    queue<int> q; // FIFO জন্য queue ব্যবহার করা হয়েছে
    bool inFrame[1000] = {0}; // পেজগুলো ফ্রেমে আছে কিনা চেক করার জন্য

    // প্রতিটি পেজ রেফারেন্স এর জন্য চেক করা
    for(int i=0; i<pages; i++){

        // পেজ যদি আগেই ফ্রেমে থাকে তাহলে Hit হবে
        if(inFrame[ref[i]]){
            hits++;
        }
        else{
            // যদি ফ্রেম ফুল হয়ে থাকে তাহলে প্রথম পেজ বের করে দেওয়া (FIFO)
            if(q.size() == frames){
                int old_page = q.front(); // প্রথম পেজ
                q.pop();                  // প্রথম পেজ বের করা
                inFrame[old_page] = false; // সেই পেজকে "ফ্রেমে নেই" বলে মার্ক
করা
            }

            // নতুন পেজটি ফ্রেমে ঢোকানো
            q.push(ref[i]);
            inFrame[ref[i]] = true; // নতুন পেজকে "ফ্রেমে আছে" বলে মার্ক করা
        }
    }
```

```
    // মোট Hit ও Fault সংখ্যা প্রিন্ট করা
    cout << "\nHits: " << hits << endl;
    cout << "Faults: " << pages - hits << endl;

    return 0;
}
```

6)page replacement-LRU

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

int main(){
    int pages, frames, hits=0;

    cout<<"Enter number of pages: ";
    cin>>pages;
    int ref[pages];

    cout<<"Enter reference string:\n";
    for(int i=0; i<pages; i++)
        cin>>ref[i];

    cout<<"Enter frame size: ";
    cin>>frames;

    list<int> frame; // LRU এর জন্য list ব্যবহার
    unordered_map<int, list<int>::iterator> pos;

    for(int i=0;i<pages;i++){
        if(pos.find(ref[i])!=pos.end()){
            hits++;
            frame.erase(pos[ref[i]]); // আগের পজিশন থেকে সরিয়ে সামনের দিকে
আনা
        }
        else if(frame.size()==frames){
            pos.erase(frame.back()); // সবচেয়ে পুরানো পেজ সরানো
            frame.pop_back();
        }
        frame.push_front(ref[i]);   // নতুন পেজকে সামনে ঢোকানো
        pos[ref[i]]=frame.begin();  // পেজের নতুন পজিশন আপডেট করা
    }

    cout<<"\nHits: "<<hits<<"\nFaults: "<<pages-hits<<endl;
}
```

7)page replacement-MRU

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

int main(){
    int pages, frames, hits=0;

    cout<<"Enter number of pages: ";
    cin>>pages;
    int ref[pages];

    cout<<"Enter reference string:\n";
    for(int i=0; i<pages; i++)
        cin>>ref[i];

    cout<<"Enter frame size: ";
    cin>>frames;
    vector<int> frame;
    for(int i=0; i<pages; i++){
        bool found = false;
        for(int j : frame){
            if(j==ref[i]){
                hits++; found=true; break;
            }
        }
        if(found){
            // MRU à¦œà¦¨à§□à¦¯ à¦¸à¦®à§□à¦ªà§□à¦°à¦¤à¦¿ à¦ªà§‡à¦œ
à¦¶à§‡à¦·à§‡ à¦°à¦¾à¦–à¦¾
            for(int k=0; k<frame.size(); k++){
                if(frame[k]==ref[i]){
                    frame.erase(frame.begin()+k);
                    frame.push_back(ref[i]);
                    break;
                }
            }
            continue;
        }

        if(frame.size() < frames)
            frame.push_back(ref[i]);  // à¦«à§□à¦°à§‡à¦®à§‡
à¦œà¦¾à¦¯à¦—à¦¾ à¦¥à¦¾à¦•à¦²à§‡ à¦¢à§□à¦•à¦¾à¦"
        else{
            frame.pop_back();    // MRU à¦ªà§‡à¦œ (à¦¶à§‡à¦· à¦ªà§‡à¦œ)
à¦¸à¦°à¦¾à¦"
            frame.push_back(ref[i]);
        }
    }

    cout<<"\nHits: "<<hits<<"\nFaults: "<<pages-hits<<endl;
}
```