Ques. Write a program for implement the priority scheduling algorithm.

```
#include <iostream>
#include <stdbool.h>
#include <algorithm>
using namespace std;
class DataDetails
{
public:
   int ari, pno, bur, pri;
   int ct, tat, wt;
   bool visit;
};
bool comparator(DataDetails d1, DataDetails d2)
{
   if (d1.pri != d2.pri)
      return (d1.pri < d2.pri);
   else
   {
      if (d1.ari != d2.ari)
         return (d1.ari < d2.ari);
      return (d1.pno < d2.pno);
   }
}
bool comparatorPno(DataDetails d1, DataDetails d2)
{
   return (d1.pno < d2.pno);
}
int main()
{
   int n, ti = 0, ch = 0;
   cout << "input n:";
   cin >> n;
   float avgtat = 0, avgwt = 0;
   vector<DataDetails> arr(n);
   for (int i = 0; i < n; i++)
   {
      arr[i].pno = i;
      cin >> arr[i].ari;
      cin >> arr[i].bur;
      cin >> arr[i].pri;
   }
   for (int i = 0; i < n; i++)
      arr[i].visit = false;
   sort(arr.begin(), arr.end(), comparator);
   for (int i = 0; i < n; i++)
   {
      ch = 0;
```

```
        for (int j = 0; j < n; j++)
        {
            if (!arr[j].visit && ti >= arr[j].ari)
            {
                ti += arr[j].bur;
                arr[j].ct = ti;
                arr[j].tat = arr[j].ct - arr[j].ari;
                arr[j].wt = arr[j].tat - arr[j].bur;
                avgtat += arr[j].tat;
                avgwt += arr[j].wt;
                arr[j].visit = true;
                ch = 1;
                break;
            }
        }
        if (!ch)
        {
            ti++;
            i--;
        }
    }
    sort(arr.begin(), arr.end(), comparatorPno);
    cout << endl;
    cout << " *********SOLUTION" << endl;
    cout << "PN "
         << "AT "
         << "BT "
         << "CT "
         << "TAT "
         << "WT " << endl;
    for (int i = 0; i < n; i++)
        cout << "P" << arr[i].pno << " " << arr[i].ari << " " << arr[i].bur << " " << arr[i].ct << " " <<
arr[i].tat << " " << arr[i].wt << " " << endl;
    avgtat = avgtat / n;
    avgwt = avgwt / n;
    cout << "Average TurnAroundTime is :" << avgtat << endl;
    cout << "Average WaitingTime is :" << avgwt << endl;
    return 0;
}
```

Output Clear

```
$ input n:5
3 9 3
5 3 2
0 2 1
5 4 4
4 3 3

 *********SOLUTION****
PN AT BT CT TAT WT
P0 3  9  12  9  0
P1 5  3  15  10 7
P2 0  2  2   2  0
P3 5  4  22  17 13
P4 4  3  18  14 11
Average TurnAroundTime is :10.4
Average WaitingTime is :6.2
```

Ques. Write a program for implement the LRU (Least Recently Used) Page replacement Algorithm.

```
#include <iostream>
#include <vector>
using namespace std;
int minimumindex(vector<pair<int, int>> v)
{
    int size = v.size(), mini = v[0].second, index = 0;
    for (int i = 0; i < size; i++)
    {
        if (mini > v[i].second)
        {
            mini = v[i].second;
            index = i;
        }
    }
    return index;
}
int main()
{
    int n;
    cout << "queue size :";
    cin >> n;
    vector<pair<int, int>> v(n);
    for (int i = 0; i < n; i++)
        v[i].first = -1;
    int np, j = 0, pos = 0, count = 0;
    cout << "proce no :";
    cin >> np;
    vector<int> p(np);
    cout << "input process." << endl;
    for (int i = 0; i < np; i++)
        cin >> p[i];
    for (int i = 0; i < np; i++)
    {
        if (pos < n)
        {
            int found = 0;
            for (int k = 0; k < n; k++)
            {
                if (p[i] == v[k].first)
                {
                    found = 1;
                    v[k].second = j;
                    j++;
                    break;
                }
```

```
            }
            if (!found)
            {
                v[pos].first = p[i];
                v[pos].second = j;
                j++;
                pos++;
                count++;
            }
        }
        else
        {
            int mini = minimumindex(v);
            int found = 0;
            for (int k = 0; k < n; k++)
            {
                if (p[i] == v[k].first)
                {
                    found = 1;
                    v[k].second = j;
                    j++;
                    break;
                }
            }
            if (!found)
            {
                int index = minimumindex(v);
                v[index].first = p[i];
                v[index].second = j;
                j++;
                count++;
            }
        }
    }
    cout << "No. of hits is :" << np - count << endl;
    cout << "No. of misses is :" << count;
    return 0;
}
```

Output Clear

```
queue size :4
proce no :13
input process.
7 0  1 2 0 3 0 4 2 3 0 3 2
No. of hits is :7
No. of misses is :6
```

Ques. Write a program for implement the MRU (Most Recently Used) Page replacement Algorithm.

```
#include <iostream>
#include <vector>
using namespace std;
int maximumindex(vector<pair<int, int>> v)
{
    int size = v.size(), maxi = v[0].second, index = 0;
    for (int i = 0; i < size; i++)
    {
        if (maxi < v[i].second)
        {
            maxi = v[i].second;
            index = i;
        }
    }
    return index;
}
int main()
{
    int n;
    cout << "queue size :";
    cin >> n;
    vector<pair<int, int>> v(n);
    for (int i = 0; i < n; i++)
    {
        v[i].first = -1;
        v[i].second = -1;
    }
    int np, j = 0, pos = 0, count = 0;
    cout << "proce no :";
    cin >> np;
    vector<int> p(np);
    cout << "input process" << endl;
    for (int i = 0; i < np; i++)
        cin >> p[i];
    for (int i = 0; i < np; i++)
    {
        if (pos < n)
        {
            int found = 0;
            for (int k = 0; k < n; k++)
            {
                if (p[i] == v[k].first)
                {
                    found = 1;
                    v[k].second = j;
```

```
                j++;
                break;
            }
        }
        if (!found)
        {
            v[pos].first = p[i];
            v[pos].second = j;
            j++;
            pos++;
            count++;

        }
    }
    else
    {
        int mini = maximumindex(v);
        int found = 0;
        for (int k = 0; k < n; k++)
        {
            if (p[i] == v[k].first)
            {
                found = 1;
                v[k].second = j;
                j++;
                break;
            }
        }
        if (!found)
        {
            int index = maximumindex(v);
            v[index].first = p[i];
            v[index].second = j;
            j++;
            count++;

        }
    }
  }
  cout << "No. of hits is :" << np - count << endl;
  cout << "No. of misses is :" << count;
  return 0;
}
```

Output Clear

```
$ queue size :4
proce no :24
input process
1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6
No. of hits is :12
No. of misses is :12
```

Ques. Write a program for implement the Optimal Page replacement Algorithm.

```
#include <iostream>
#include <vector>
#include <stdbool.h>
using namespace std;
bool hit(vector<int> &arr, int key)
{
    int n = arr.size();
    for (int i = 0; i < n; i++)
        if (arr[i] == key)
            return true;
    return false;
}
int rightmost(vector<int> &f, vector<int> &p, int k)
{
    int nf = f.size(), np = p.size();
    int index = 0, maxindex = 0, ipos = 0, j;
    for (int i = 0; i < nf; i++)
    {
        for (j = k + 1; j < np; j++)
        {
            if (f[i] == p[j])
            {
                index = j;
                break;
            }
        }
        if (j == np)
            return i;
        else
        {
            if (maxindex < index)
            {
                maxindex = index;
                ipos = i;
            }
        }
    }
    return ipos;
}
int main()
{
    int fs, pno, count = 0, j = 0;
    cout << "frame size :";
    cin >> fs;
    vector<int> f(fs, -1);
```

```
    cout << "input no of processor :";
    cin >> pno;
    vector<int> p(pno);
    for (int i = 0; i < pno; i++)
        cin >> p[i];
    for (int i = 0; i < pno; i++)
    {
        if (j < fs)
        {
            bool found = hit(f, p[i]);
            if (!found)
            {
                f[j] = p[i];
                j++;
                count++;
            }
        }
        else
        {
            bool found = hit(f, p[i]);
            if (!found)
            {
                int index;
                index = rightmost(f, p, i);
                f[index] = p[i];
                count++;
            }
        }
    }
    cout << "No. of hits is :" << pno - count << endl;
    cout << "No. of misses is :" << count;
    return 0;
}
```

Output Clear

```
frame size :4
input no of processor :14
7 0 1 2 0 3 0 4 2 3 0 3 2 3
No. of hits is :8
No. of misses is :6
```

Ques. Write a C program for implementing the PIPE.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main()
{
   int fd[2], n;
   char buffer[100];
   pid_t p;
   pipe(fd);
   p = fork();
   if (p > 0)
   {
      printf("Parent Passing value to child.\n");
      write(fd[1], "hello\n", 6);
      wait(NULL);
   }
   else
   {
      printf("Child printing received value\n");
      n = read(fd[0], buffer, 100);
      write(1, buffer, n);
   }
}
```

Output Clear

```
$ ┌──(root㉿DELL)-[/mnt/c/Users/mohdn/Desktop/Lab/Operating System]
└─# gcc pipe.c -o pipe

┌──(root㉿DELL)-[/mnt/c/Users/mohdn/Desktop/Lab/Operating System]
└─# ./pipe
Parent Passing value to child.
Child printing received value
hello
```