

**INTERNAL ASSIGNMENT - JUL2023 **

INTERNAL ASSIGNMENT

**Name: Deepankar Sharma**

**Student ID: 233512013**

**Course Code: OMC-109**

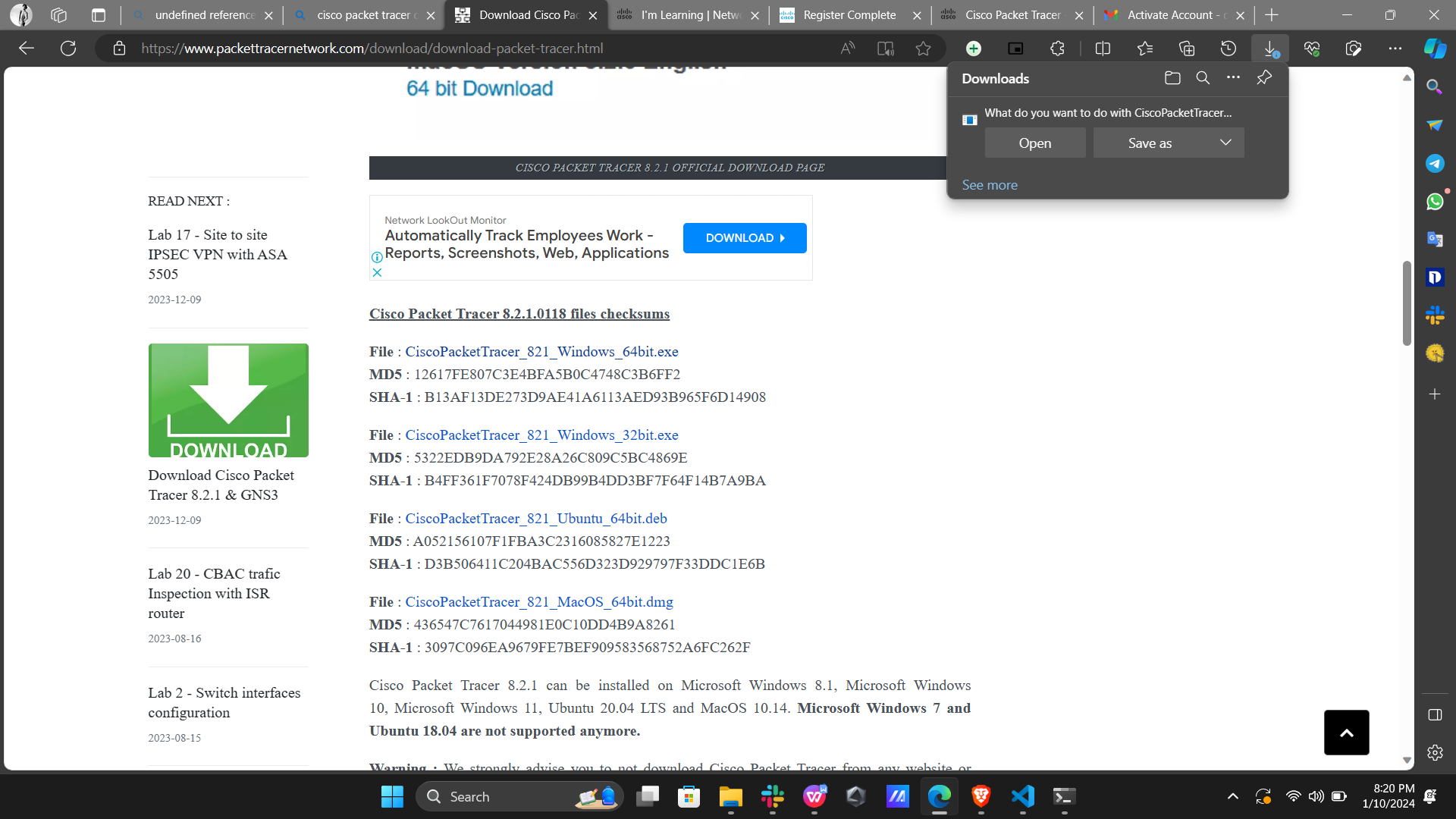
**Course Title: Operating Systems & Computer Networks lab**

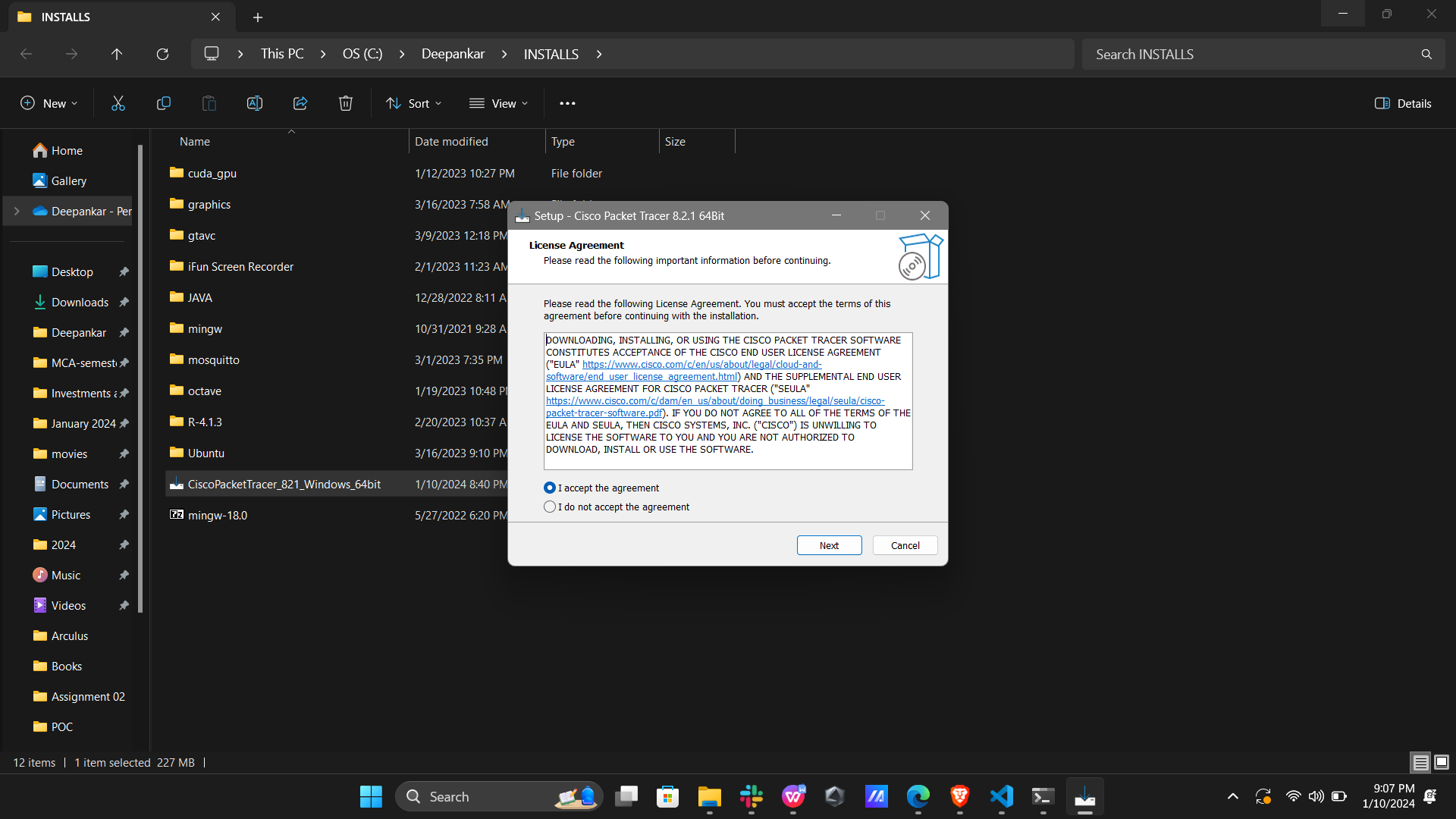
|  |  |
| --- | --- |
| **Q.No.** | **Question** |
| 1 | Write a C program to Simulate the following Memory management algorithm-First fit  # include <stdio.h>  *void* first\_fit(*int* *m*, *int* *n*, *int* *Blocks*[], *int* *Process*[]){  *int* i,j;  *int* allocation[*n*];      for ( i = 0; i < *n*; i++)      {          /\* code \*/          allocation[i]= -1;      }      for ( i = 0; i < *n*; i++) // # processes      {          /\* code \*/          for (  j = 0; j<*m*; j++) // # blocks          {              /\* code \*/              if (*Blocks*[j] >=*Process*[i]){                  allocation[i]= j;  *Blocks*[j]= *Blocks*[j]-*Process*[i];                  break;                }          }        }      printf("\nP. No.\tP. Size\tBlock No.\n");      for (i = 0; i < *n*; i++)      {          /\* code \*/          printf("%d\t%d\t", i+1, *Process*[i]);          if (allocation[i]!=-1)          {              printf("%i\n", allocation[i]+1);          }else printf("Not Allocated\n");        }  }  *int* main(){  *int* m, n, Blocks[10], Process[10];      printf("Enter # processes: "); scanf("%d", &n);      printf("Enter # blocks: "); scanf("%d", &m);      printf("Enter the process sizes\n");      for (*int* i = 0; i < n; i++)      {          scanf("%d", &Process[i]);      }      printf("Enter the block sizes\n");      for (*int* i = 0; i < m; i++)      {          scanf("%d", &Blocks[i]);      }      first\_fit(m, n, Blocks, Process);      return 0;    } |
| 2 | Write a C program to Implement the optimal page replacement algorithm  #include<stdio.h>  *int* main() {  *int* num\_frames, num\_pages, frames[10], pages[30], temp[10];  *int* flag1, flag2, flag3, i, j, k, pos, max, faults=0, hit;      printf("Enter #frames, #pages\n");  scanf("%d%d", &num\_frames, &num\_pages);      printf("Enter page reference string: \n");      for (i = 0; i < num\_pages; i++) {          scanf("%d", &pages[i]);  }      for (i = 0; i < num\_frames; i++) {          frames[i] = -1;  }      for (i = 0; i < num\_pages; i++) {          flag1 = flag2 = 0;          hit = 0;          for (j = 0; j < num\_frames; j++) {              if (frames[j] == pages[i]) {                  flag1 = flag2 = 1;                  hit = 1;                  break;              }          }          if (flag1 == 0) {              for (j = 0; j < num\_frames; j++) {                  if (frames[j] == -1) {                      faults++;                      frames[j] = pages[i];                      flag2 = 1;                      break;                  }              }          }          if (flag2 == 0) {              flag3 = 0;              for (j = 0; j < num\_frames; j++) {                  temp[j] = -1;                  for (k = i + 1; k < num\_pages && temp[j] == -1; k++) {                      if (frames[j] == pages[k]) {                          temp[j] = k;                      }                  }              }              for (j = 0; j < num\_frames; j++) {                  if (temp[j] == -1) {                      pos = j;                      flag3 = 1;                      break;                  }              }              if (flag3 == 0) {                  max = temp[0];                  pos = 0;                  for (j = 1; j < num\_frames; j++) {                      if (temp[j] > max) {                          max = temp[j];                          pos = j;                      }                  }              }              frames[pos] = pages[i];              faults++;          }          if (hit == 0) {              printf("\n");              for (j = 0; j < num\_frames; j++) {                  printf("%d\t", frames[j]);              }          }  }      printf("\n\nTotal Page Faults= %d\n", faults);      return 0;  } |
| 3 | Implement a program in C to extract process ID (PID) and parent process ID (PPID)  #include<stdio.h>  #include<stdlib.h>  #include<unistd.h>  *int* main(){  *int* pid;      pid= fork();      if (pid==-1){          perror("fork failed");          exit(0);      }      else if (pid==0){          printf("\n1.1 Child process is under execution");          printf("\n1.2 Process ID of Child process is [%d]", getpid());          printf("\n1.3 Process ID of Parent process is [%d]\n", getppid());      }      else{          printf("\n2.1 Parent process is under execution");          printf("\n2.2 Process ID of Parent process is [%d]", getpid());          printf("\n2.3 Process ID of Parent process is [%d]\n", getppid());      }      return 0;  } |
| 4 | Simulate the following CPU scheduling algorithms-FCFS  #include<stdio.h>  #include<stdlib.h>  *struct* Process  {      /\* data \*/  *int* pid;  *int* bt;  *int* at;  };  *void* fcfs\_scheduling(*struct* Process\**proc*, *int* *n*){  *int* i, wt[*n*], tat[*n*], total\_wt=0, total\_tat=0;      // calculate waiting time for each process      wt[0]= 0;      for ( i = 1; i < *n*; i++)      {          /\* code \*/          wt[i]= wt[i-1]+ *proc*[i-1].bt;      }      // calculate turnaround time for each process      for ( i = 0; i < *n*; i++)      {          /\* code \*/          tat[i]= wt[i]+ *proc*[i].bt;      }      // calculate total waiting and turnaround time      for (i = 0; i < *n*; i++)      {          /\* code \*/          total\_wt+=wt[i];          total\_tat+=tat[i];      }      printf("\nPID\tBT\tAT\tWT\tTAT\n");      for ( i = 0; i < *n*; i++)      {          /\* code \*/          printf("%d\t%d\t%d\t%d\t%d\n", *proc*[i].pid, *proc*[i].bt, *proc*[i].at, wt[i], tat[i]);      }      printf("\nAverage waiting time: %.2f\n", (*float*)total\_wt/*n*);      printf("\nAverage turnaround time: %.2f\n", (*float*)total\_tat/*n*);  }  *int* main(){  *int* n, i;      printf("Enter the number of processes: ");      scanf("%d", &n);  *struct* Process proc[n];      for ( i = 0; i < n; i++)      {          /\* code \*/          printf("Enter the burst time and arrival time for process %d: ", i+1);          scanf("%d%d", &proc[i].bt, &proc[i].at);          proc[i].pid= i+1;      }      fcfs\_scheduling(proc, n);      return 0;  } |

**Compulsory question:**

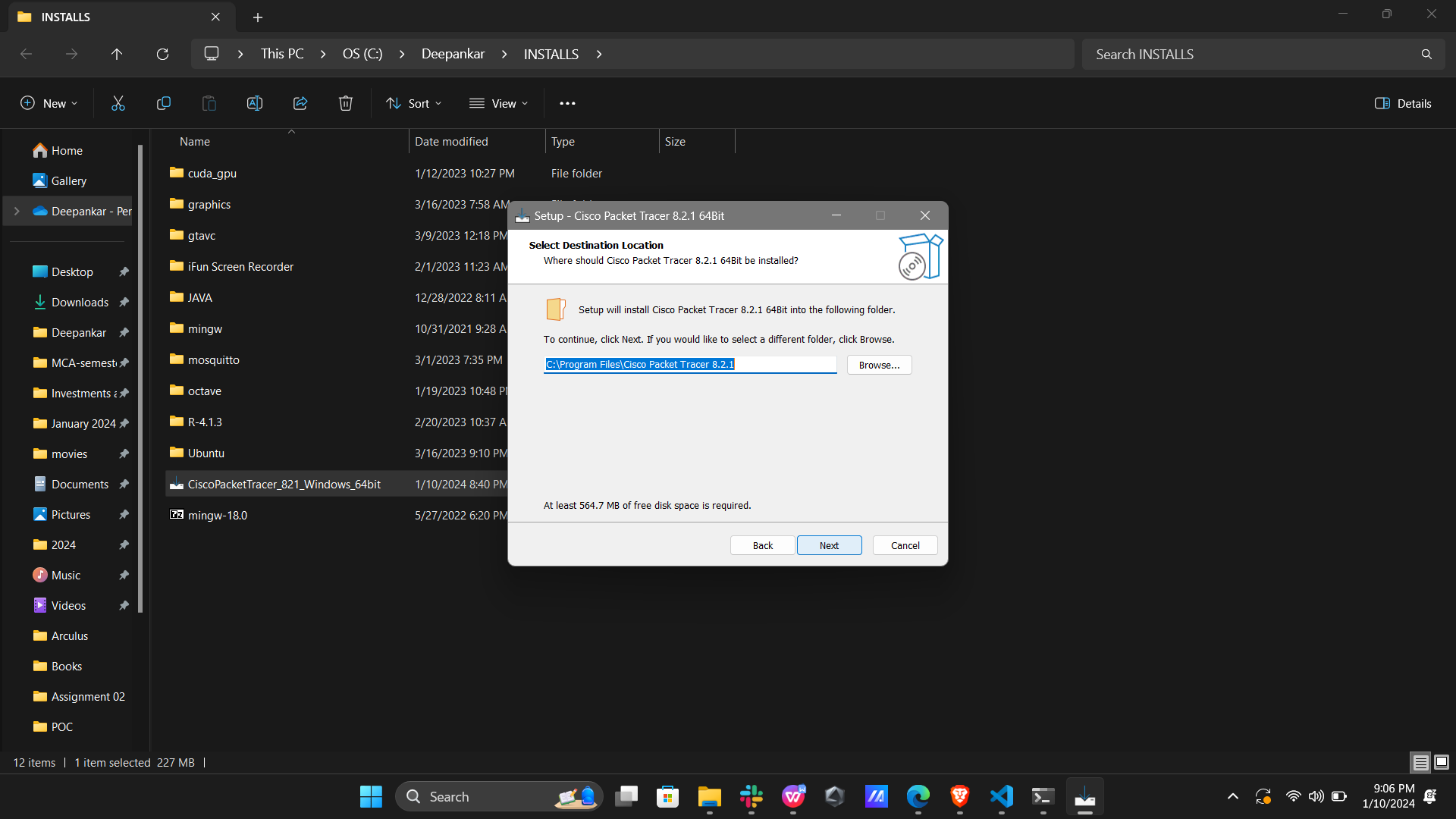
Explain the installation steps for Cisco Packet Tracer, and include snapshots for clarification.

1. Download the latest version of Cisco Packet Tracer from the official website or from your instructor.



2. Run the installer file and accept the license agreement.

3. Choose the installation location and the components you want to install. You can also customize the shortcuts and associations.



4. Click on Install and wait for the installation to complete.

