```c
#include"header.h"

int Pack(char* path , char* comb )
{
        DIR *dirp = NULL;
        struct dirent * filep = NULL;
        int fd =0 ;
        char Authenticate[4] = "123";
        int Key =1234;

// OPENING DIRECTORY
        dirp = opendir( path );
        if(dirp == NULL)
        {
                printf("ERROR 5 : Failed to open %s path \n" , path );
                return 0;
        }

// CREATING PACKED FILE
        fd = creat(comb , 0777);
        if(fd == -1)
        {
                printf("ERROR 6 : Failed to Create %s file \n" , comb );
                return 0;
        }

        write(fd , Authenticate , 4);

// MERGING FILES
        while((filep = readdir( dirp )) != NULL)
        {
                int fd1 = 0;
                int ret = 0;
                char FileName[200];
                struct stat buf;
                struct info file ;
                char temp[2];

        //IGNORING . AND .. FILES
                strcpy(temp,filep->d_name);
                if (  temp[0] == '.' )
                continue;

                sprintf(FileName,"%s/%s",path,filep->d_name);

                ret = stat(FileName, &buf);
                if(ret == 1)
                {
                        printf("ERROR 7 : Failed to open %s file from directory\…
                        return 0;
                }

                if(S_ISREG(buf.st_mode))
                {
                        strcpy(file.name , filep->d_name);
                        file.size = buf.st_size;

                        fd1 = open(FileName , O_RDONLY);
                        char (* Data)[file.size] = (char (*)[])malloc(file.size …
                        read(fd1 , Data , file.size);

                // ENCRYPTION
```

```
63                          for(int i = 0 ; i < file.size ; i++)
64                          {
65                                  Key -= i;
66                                  (*Data)[i] +=  Key ;
67                          }
68
69                          write(fd , (char *)&file , sizeof(file) );
70                          write(fd , Data , file.size);
71                  }
72          }
73          return 1;
74  }
75
76  ////////////////////////////////////////////////////////////////////////////…
77  int Unpack(char* comb , char* dir )
78  {
79          char name[200] ;
80          int ret = 0 , fd = 0 , auth = 0 ;
81          char Authenticate[4];
82          int Key =1234;
83
84  // CREATING NEW DIRECTORY AS UNPACKED_FILES
85          sprintf(name , "%s/UNPACKED_FILES",dir);
86          ret = mkdir(name , 0777);
87          if(ret == 1)
88          {
89                  printf("ERROR 8 : Failed to Create new directory %s \n" , name );
90                  return 0;
91          }
92
93  // OPENING PACKED FILE
94          fd = open(comb , O_RDONLY);
95          if(fd == -1)
96          {
97                  printf("ERROR 9 : Failed to open %s File\n" , comb );
98                  return 0;
99          }
100
101 // AUTHENTICATION OF PROVIDED PACKED FILE
102         read(fd , Authenticate , 4);
103         auth = strcmp(Authenticate , "123");
104         if(auth != 0)
105         {
106                 printf("ERROR 10 : Provide approciate packed file\n");
107                 return 0;
108         }
109
110 // CREATING FILES AND COPYING DATA
111         while(1)
112         {
113                 struct info file;
114                 char path[200];
115                 int fd1 = -1;
116
117                 ret= read(fd , &file , sizeof(file));
118
119                 if(ret == 0)
120                         break;
121
122                 sprintf(path,"%s/%s",name,file.name);
123                 char (* Data)[file.size] = (char (*)[])malloc(file.size * sizeof…
124
```

```
125                     fd1 = creat(path , 0777);
126                     read(fd , Data , file.size);
127
128             // DECRYPTION
129                     for(int i = 0 ; i < file.size ; i++)
130                     {
131                             Key -= i;
132                             (*Data)[i] -= Key;
133                     }
134
135                     write(fd1 , Data , file.size);
136
137             }
138             return 1;
139     }
```