PRANAV JAGADEESH 1BM18CS071 5B BATCH 4

PROGRAM 8 : FUNCTIONS OF DICTIONARY USING HASHING

```
typedef struct list
{ int data
  struct list *next
} node_type
node_type *ptr[max], *root[max], *temp[max]
class Dictionary
{ public: int index
          Dictionary ()
          void insert(int)
          void search(int)
          void delete_ele(int)
};
Dictionary :: Dictionary ()
{ index = -1
  for (int i = 0, i<max, i++)
  { root[i] ← NULL
    ptr[i] ← NULL
    temp[i] ← NULL
  }
}
void Dictionary :: insert(int Key)
{ index ← int(Key % max)
  ptr[index] ← (node_type *) malloc(sizeof(node_type))
  ptr[index] → data ← Key
```

```
if ( root [index] <=> NULL)
{  root [index] <- ptr [index]
   root [index] -> next <- NULL
   temp [index] <- ptr [index]
}
else { temp [index] <- root [index]
       while ( temp [index] -> next != NULL)
       temp [index] <- temp [index] -> next
       temp [index] -> next <- ptr [index]
     }
}
void Dictionary :: search ( int key)
{ int flag <- 0
  index <- int ( key % max)
  temp [index] <- root [index]
  while ( temp [index] -> next != NULL)
       temp [index] <- temp [index] -> next
       temp [index] -> next <- ptr [index]
  }
}
void Dictionary :: search ( int key)
{ int flag <- 0
  index <- int ( key % max)
  temp [index] <- root [index]
  while [ temp [index] != NULL)
  { if ( temp [index] -> data <=> key)
    { print  key is found
      flag <- 1
      break
    }
  else temp [index] <- temp [index] -> next
}
```

```
if (flag <-> 0)
   print "Key not found"
}

void Dictionary :: delete_ele (int key)
{ index <- int (key % max)
  temp [index] <- root [index]
  while (temp [index] -> data |<-) key && temp[index] |<- NULL)
  { ptr [index] <-) temp [index]
    temp [index] <-) temp [index] -> next
  }
  ptr [index] -> next <- temp [index] -> next
  print temp [index] -> data
  temp [index] -> data <- -1
  temp [index] <-) NULL
   free (temp [index])
}

main()
{ int val, ch, n, num
   char c
   Dictionary d
   do
   {
      switch (ch)
      case 1: &i <- value
              for int i <-0, i<n, i++ .{ num <- value
                 d.insert (num)
              } break
      case 2: n <- value
              d. search (n)
      case 3: n <- value
              d.delete_ele (n)
              break
```

```
default : print Invalid choice
}
print y to continue
'c <- value
}
while ( c == 'y' );
getch();
}
```