รายงาน

โปรแกรม PatiTin_2

จัดทำโดย

นาย ภูริภัทร	อภิรักษ์โชติมา	65070501043
นาย มาวิน	ศรีชาติ	65070501045
นาย รอซีน	มูเก็ม	65070501047
นาย ศุภจิตต์	จันทมูล	65070501054
นาย พิชวัฒน์	อดุลย์วิทยากร	65070501080

เสนอ

ดร. ปิยนิตย์ เวปุลานนท์

ดร. ทวีชัย นันทวิสุทธิวงศ์

รายงานฉบับนี้เป็นส่วนหนึ่งของ รายวิชา CPE11203

IMPLEMENTATION OF DATA STRUCTURE IN MACHINE LEARNING

ภาคเรียนที่ 2 ปีการศึกษา 2565

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

บทนำ

พัฒนาโปรแกรมเกี่ยวกับอะไร

ณ ปัจจุบัน การวางแผนและการจัดเวลาถือเป็นสิ่งที่สำคัญในการดำเนินชีวิต หากไม่มีการวางแผนหรือการบริหารเวลาที่มีประสิทธิภาพ อาจก่อให้เกิดความสับสนและล่าช้าในการทำงาน

ด้วยเหตุนี้ กลุ่มของเราได้สร้างโปรแกรมชื่อ PatiTin_2 ขึ้นมา เพื่อช่วยให้ผู้ใช้งานสามารถจัดการวางแผนด้านเวลาของแต่ละงานที่กำหนดอย่างมีประสิทธิภาพ โดยโปรแกรม PatiTin_2 จะมีลักษณะคล้ายกับ To Do List โดยสามารถจดบันทึกงานที่ต้องทำโดยการกำหนดชื่อ วัน และเวลาของงานเพื่อให้ตัวโปรแกรมสามารถแสดงงานที่ต้องทำในวันนั้นได้ และยังสามารถกำหนดความสำคัญของแต่ละงานได้เพื่อที่จะนำไปดูว่ามีงานไหนที่สำคัญบ้าง หรือจะเรียกดูงานที่ต้องการจากการค้นหาด้วยวันที่ของงานที่ต้องทำก็ได้

นอกจากนี้ โปรแกรม PatiTin_2 ยังช่วยให้ผู้ใช้งานมีการวางแผนในการใช้เวลาได้อย่างมีประสิทธิภาพสูงสุด โดยการจัดลำดับความสำคัญของงานและกำหนดเวลาในการทำงานได้อย่างมีประสิทธิภาพ

อีกทั้งยังสามารถใช้งานโปรแกรม PatiTin_2 เพื่อเพิ่มประสิทธิภาพในการทำงานร่วมกับผู้อื่นได้อีกด้วย โดยการแชร์งานให้กับผู้อื่นในทีมหรือคนใกล้ชิด หรือการกำหนดงานร่วมกันในโครงการที่มีหลายคนเข้าร่วม ทำให้ผู้ใช้งานสามารถวางแผนและจัดการเวลาของทีมหรือโครงการได้อย่างมีประสิทธิภาพ และช่วยเพิ่มความสามารถในการทำงานร่วมกันอย่างมีประสิทธิภาพสูงสุด

หน้าที่รับผิดชอบ

65070501043 ภูริภัทร อภิรักษ์โชติมา

- รายงาน (Report)

65070501045 มาวิน ศรีชาติ

- ออกแบบฐานข้อมูล (Database)
- บันทึกข้อมูลและเรียกข้อมูล (Save file and Load file)

65070501047 รอซีน มูเก็ม

- ทดสอบระบบหาข้อผิดพลาด (Tester)
- การรับข้อมูลจากผู้ใช้ (User input)

65070501054 ศุภจิตต์ จันทมูล

- ออกแบบโครงสร้างข้อมูลและอัลกอริทึม

  (Data structure and Algorithm designer)

- ผู้เขียนโปรแกรมหลัก (Main coder)

65070501080 พิชวัฒน์ อดุลย์วิทยากร

- ออกแบบและจัดระเบียบหน้าตาของแอปพลิเคชัน

  (User Interface)

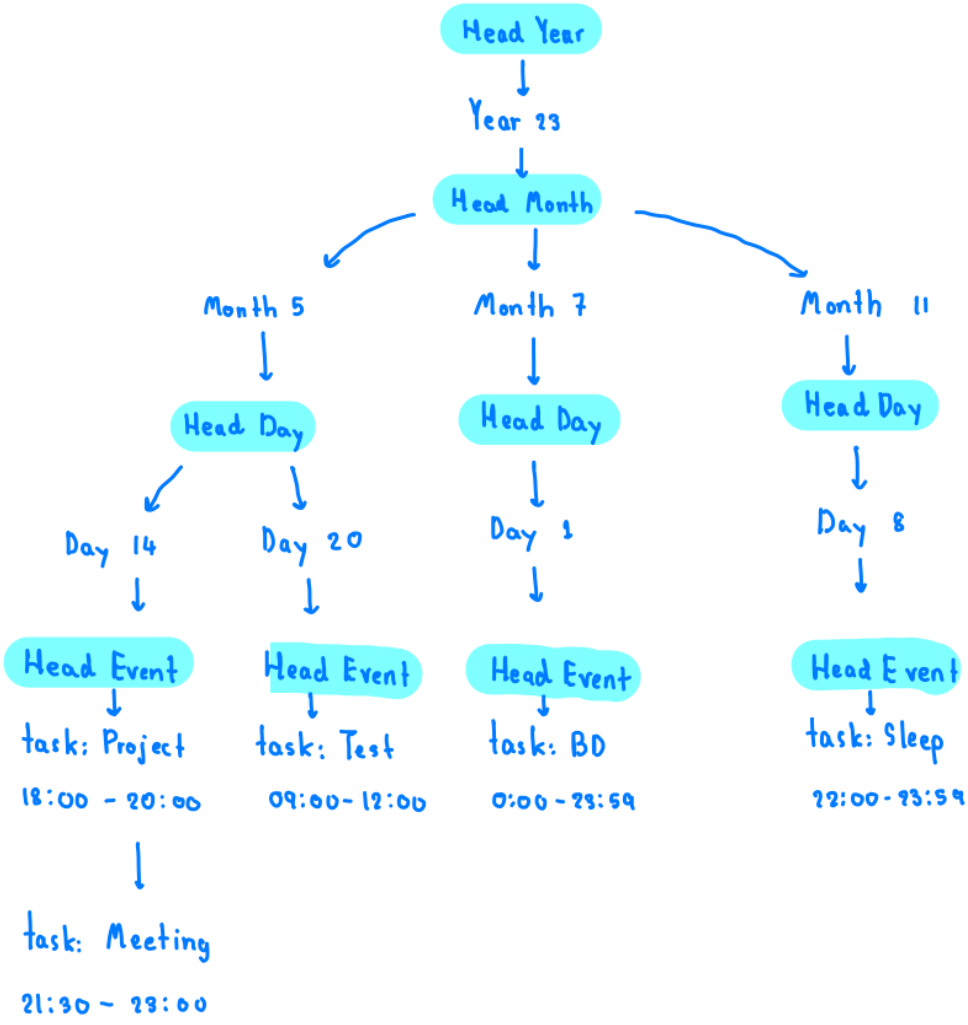
วิธีการดำเนินงาน

โครงสร้างข้อมูลที่ใช้

- Priority Queue : สำหรับการเก็บ Task ที่ต้องทำ โดยเก็บเรียงจากเวลาที่ใกล้กับเวลาปัจจุบันมากที่สุด
- Stack : สำหรับการเก็บลำดับ Task ที่จบไปแล้ว
- Linked List : สำหรับการใช้งานคู่กับ Priority Queue
- Tree (Forest) : สำหรับการเก็บข้อมูลวัน เดือน ปี และ Task
- Array : สำหรับการเก็บปี, ใช้คู่กับ Tree และจะใช้คู่กับ Stack

อัลกอริทึม

เริ่มต้นด้วยการสร้าง Forest โดยที่มีรากเป็นปี เมื่อ User ต้องการเพิ่ม Task จะต้องระบุ วัน เดือน ปี เวลา ชื่อ และความสำคัญของ Task ที่ต้องทำ โดยจะทำการจองหน่วยความจำเพิ่มเมื่อ User ยังไม่เคยมีกิจกรรมในแต่ละเดือนของปีนั้นๆ และจะทำการจองหน่วยความจำเพิ่มเมื่อไม่เคยมีกิจกรรมในวันนั้นของเดือนและปีที่เราระบุ โดยการเพิ่มวันและเดือนจะใช้ Priority Queue ในการเรียงลำดับเลขเดือนจากน้อยไปมาก และจะเรียงลำดับ Task ตามเวลาเริ่มต้นที่น้อยที่สุดก่อน หาก Task มีเวลาเท่ากันจะให้ Task ที่มาก่อนขึ้นก่อน และจะใช้ Stack เก็บ Task ที่หมดอายุไปแล้วเมื่อเทียบกับเวลาจริง



แผนภาพตัวอย่างการจัดเก็บข้อมูลของแต่ละ Task

(ในการเขียนโปรแกรมจะแปลง Tree ดังภาพเป็น Binary tree แทน)

โค้ดโปรแกรม

Struct event

```
typedef struct event {
  char *event;
  char startTime[12];
  char endTime[12];
  int favorite;
  struct event *next;
} Event;
```

Struct event จะเก็บข้อมูลเวลาเริ่ม เวลาจบเป็น String และตัวแปร favorite ที่จะเป็นตัวบ่งบอกลำดับความสำคัญของ Task นั้นๆ

Struct day

```
typedef struct day {
  int day;
  struct day *next;
  struct event *event;
} Day;
```

Struct day จะเก็บข้อมูลวันที่ของ Task เป็น Int และเก็บ Task แรกในวันนั้นๆ ไว้ด้วย

Struct month

```
typedef struct month {
  int month;
  struct day *day;
  struct month *next;
} Month;
```

Struct month จะเก็บข้อมูลเดือนของ Task เป็น Int และเก็บวันที่ของ Task แรกในเดือนนั้นๆ ไว้ด้วย

Struct year

```
typedef struct year {
  int year;
  struct month *month;
  struct year *next;
} Year;
```

Struct year จะเก็บข้อมูลปีของ Task เป็น Int และเก็บเดือนของ Task แรกในปีนั้นๆ ไว้ด้วย

Struct stack

```
typedef struct stack {
  int top;
  int size;
  Year *arr[100];
} Stack;
```

Struct stack จะเก็บตัวแปร top ที่เป็น index ของ Task ล่าสุดที่เลยกำหนดเวลามาแล้ว เก็บ size ของ Stack และเก็บ Struct year ซึ่งเป็นปีที่มี Task ที่เลยกำหนดเวลามาแล้ว

createStack ()

```
//initialize stack
Stack *createStack(int size) {
  Stack *stack = (Stack *)malloc(sizeof(Stack));
  stack->top = -1;
  stack->size = size;
  return stack;
}
```

ฟังก์ชัน createStack ทำหน้าที่สร้าง Stack เปล่าเพื่อนำมาเก็บ Task ที่เลยกำหนดเวลามาแล้ว

push ()

```
//append data at top of stack
void push(Stack *stack, Year *curYear) {
  if (stack->top == stack->size - 1) {
    printf("Error: Stack is full!\nYou must delete
some outdated data\n");
    return;
  }
  stack->arr[++stack->top] = curYear;
}
```

ฟังก์ชัน push ทำหน้าที่ push Struct year ลงใน Stack

pop ()

```
//iterate to second top of stack
void pop(Stack *stack) {
  if (stack->top == -1) {
    printf("Error: Something went wrong.!\n");
    return;
  }
  stack->top--;
}
```

ฟังก์ชัน pop ทำหน้าที่ pop Struct year ออกจาก Stack

peek ()

```
//return top data of stack
Year *peek(Stack *stack) {
  if (stack->top == -1) {
    printf("No outdated event yet\n");
    return NULL;
  }
  return stack->arr[stack->top];
}
```

ฟังก์ชัน peek ทำหน้าที่ return top จาก Stack

trim ()

```
//Clear whitespace at begin and tail of the text
void trim(char *str) {
  int len = strlen(str);
  int st = 0, en = len - 1;
  //create two pointer
  //isspace-> (if(str[i]==' ')) include from ctype.h
  while (isspace(str[st])) {
    st++;
  }
  while (en > st && isspace(str[en])) {
    en--;
  }
  //mark the end of string as NULL then use memmove to
copy a block of memory from a location to another
  str[en + 1] = '\0';
  memmove(str, str + st, en - st + 2);
}
```

ฟังก์ชัน trim ทำหน้าที่ลบ white-space ด้านหน้าและด้านหลังของ String

enterRemover ()

```c
//remove enter at the last of string
void enterRemover(char *str) {
  if (str[strlen(str) - 1] == '\n') {
    str[strlen(str) - 1] = '\0';
  }
}
```

ฟังก์ชัน enterRemover ทำหน้าที่ลบการเว้นบรรทัดที่เกิดจากการใช้ fgets

isOutdatedData ()

```
int isOutdatedData(int year, int month, int day, char *eventTime) {
  int yearTime, monthTime, dayTime;
  char nowTime[12];
  time_t current_time;
  struct tm *time_info;
  char date_string[20];

  // Get the current time
  current_time = time(NULL);

  // Convert the current time to local time
  time_info = localtime(&current_time);

  // Format the date string
  strftime(date_string, sizeof(date_string), "%d/%m/%y", time_info);
  strftime(nowTime, sizeof(nowTime), "%H:%M", time_info);

  sscanf(date_string, "%d/%d/%d", &dayTime, &monthTime, &yearTime);

  unsigned int status = 0;
  if ((year < yearTime)) {
    status = 1;
  } else if ((year == yearTime) && (month < monthTime)) {
    status = 1;
  } else if ((year == yearTime) && (month == monthTime) && (day <
dayTime)) {
    status = 1;
  } else if ((year == yearTime) && (month == monthTime) && (day ==
dayTime) &&
             (strcmp(eventTime, nowTime) == -1)) {
    status = 1;
  }
  return status;
}
```

ฟังก์ชัน isOutdatedData ทำหน้าที่เช็กว่า Task เลยกำหนดเวลามาแล้วหรือยัง โดยรับวัน เดือน ปี และเวลาของ Task ที่ต้องการเช็ก มาเทียบกับเวลาจริง ถ้าเวลาจริงมากกว่า (Task เลยกำหนดเวลามาแล้ว) return 1 แต่ถ้าเวลาของ Task มากกว่า (Task ยังไม่เลยกำหนดเวลา) return 0

findDateBlank ()

```
//check that user fill -- or not when input in date field
int findDateBlank(char *date, int *day, int *month, int *year) {

  int ch1 = 0, ch2 = 0, ch3 = 0;
  char *dayStr = strtok(date, "/");
  char *monthStr = strtok(NULL, "/");
  char *yearStr = strtok(NULL, "\n");

  if (dayStr == NULL || monthStr == NULL || yearStr == NULL)
    return 0;

  if (strcmp(dayStr, "--") == 0) {
    *day = -1;
    ch1 = 1;
  } else {
    if (sscanf(dayStr, "%d", day) == 1)
      ch1 = 1;
  }

  if (strcmp(monthStr, "--") == 0) {
    *month = -1;
    ch2 = 1;
  } else {
    if (sscanf(monthStr, "%d", month) == 1)
      ch2 = 1;
  }

  if (strcmp(yearStr, "--") == 0) {
    *year = -1;
    ch3 = 1;
  } else {
    if (sscanf(yearStr, "%d", year) == 1)
      ch3 = 1;
  }

  if (ch1 && ch2 && ch3)
    return 1;
  else
    return 0;
}
```

ฟังก์ชัน findDateBlank ทำหน้าที่เช็กว่าข้อมูลวันเดือนปีที่ User กรอกเป็น “--” ไหม โดยจะแบ่งข้อมูลวันเดือนปีเป็น 3 ส่วน แล้วเช็กทีละอันว่าข้อมูลเป็น “--” ไหม ถ้าใช่ก็จะให้ข้อมูลนั้นมีค่าเป็น -1 และถ้า User กรอกข้อมูลตรงตามรูปแบบ ฟังก์ชั่นจะ return 1 แต่ถ้าไม่ ฟังก์ชันจะ return 0

findTimeBlank ()

```
//check that user fill -- or not when input in time field
int findTimeBlank(char *time) {

  int hour, minute;

  if (!strcmp(time, "--"))
    return 1;

  if (sscanf(time, "%d:%d", &hour, &minute) != 2) {

    return 0;
  }
  // validate hour and minute
  if (hour < 0 || hour > 23 || minute < 0 || minute > 59)
{

    return 0;
  }

  return 1;
}
```

ฟังก์ชัน findTimeBlank ทำหน้าที่เช็กว่าข้อมูลเวลาที่ User กรอกเป็น “--” ไหม โดยถ้าเป็น “--” หรือข้อมูลตรงตามรูปแบบ ฟังก์ชันจะ return 1 กรณีอื่นๆ ฟังก์ชันจะ return 0

findFavBlank ()

```
//check that user fill -- or not when input in favorite
field
int findFavBlank(char *buffer, int *fav) {

  if (strcmp(buffer, "--") == 0) {
    *fav = -2;
    return 1;
  } else {
    if (sscanf(buffer, "%d", fav) == 1) {
      if (*fav < -1) {
        return 0;
      } else {
        return 1;
      }
    }
  }
  return 0;
}
```

ฟังก์ชัน findFavBlank ทำหน้าที่เช็กว่าข้อมูลค่าความสำคัญที่ User กรอกเป็น “--” ไหม ถ้าใช่ จะให้ค่าความสำคัญมีค่าเป็น -2 และถ้า User กรอกข้อมูลตรงตามรูปแบบ ฟังก์ชั่นจะ return 1 แต่ถ้าไม่ ฟังก์ชันจะ return 0

validateDate ()

```c
//validate the date input from user by passing string
int validateDate(char *date) {
  int day, month, year;
  // check does user input in the right format
  if (sscanf(date, "%d/%d/%d", &day, &month, &year) != 3) {
    return 0;
  }

  // validate day month year
  if (day < 1 || day > 31 || month < 1 || month > 12 || year < 1)
    return 0;

  // check for leap year
  if (month == 2) {
    int maxFebDay = 28;
    if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
      maxFebDay = 29;
    }
    if (day < 1 || day > maxFebDay) {
      return 0;
    }
  }

  // check day in month that have 30 day
  if (month == 4 || month == 6 || month == 9 || month == 11) {
    if (day < 1 || day > 30) {
      return 0;
    }
  }

  return 1;
}
```

ฟังก์ชัน validateDate ทำหน้าที่เช็กว่าข้อมูลวันเดือนปีที่ User กรอกตรงตามรูปแบบหรือไม่ โดยแบ่งเช็กวันเดือนปีเป็น 3 ส่วน และเช็กว่าตรงกับวันเดือนปีจริงหรือไม่ ถ้าข้อมูลตรงตามรูปแบบ ฟังก์ชันจะ return 1 แต่ถ้าไม่ ฟังก์ชันจะ return 0

validateDateNum ()

```
//validate the date input from user by passing integer
int validateDateNum(int year, int month, int day) {

  // validate day month year
  if (day < 1 || day > 31 || month < 1 || month > 12 || year < 1)
    return 0;

  // check for leap year
  if (month == 2) {
    int maxFebDay = 28;
    if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
      maxFebDay = 29;
    }
    if (day < 1 || day > maxFebDay) {
      return 0;
    }
  }

  // check day in month that have 30 day
  if (month == 4 || month == 6 || month == 9 || month == 11) {
    if (day < 1 || day > 30) {
      return 0;
    }
  }

  return 1;
}
```

ฟังก์ชัน validateDateNum ทำหน้าที่เช็กว่าวันเดือนปีที่ User กรอกตรงกับวันเดือนปีจริงหรือไม่ ถ้าใช่ ฟังก์ชันจะ return 1 แต่ถ้าไม่ ฟังก์ชันจะ return 0

validateTime ()

```
//validate the time input from user
int validateTime(char *time) {

  int hour, minute;
  // check does user input in the right format
  if (sscanf(time, "%d:%d", &hour, &minute) != 2) {
    return 0;
  }
  // validate hour and minute
  if (hour < 0 || hour > 23 || minute < 0 || minute > 59)
    return 0;

  return 1;
}
```

ฟังก์ชัน validateTime ทำหน้าที่เช็กว่าข้อมูลเวลาที่ User กรอกตรงตามรูปแบบหรือไม่ โดยแบ่งเช็กเวลาเป็น 2 ส่วน (ชั่วโมงและนาที) และเช็กว่าตรงกับเวลาจริงหรือไม่ ถ้าข้อมูลตรงตามรูปแบบ ฟังก์ชันจะ return 1 แต่ถ้าไม่ ฟังก์ชันจะ return 0

createEvent ()

```
Event *createEvent(char *str, char start[], char end[], int
favorite) {
  Event *tmp = (Event *)malloc(sizeof(Event));
  tmp->event = (char *)malloc((sizeof(char) * strlen(str)) + 1);
  strcpy(tmp->event, str);
  strcpy(tmp->startTime, start);
  strcpy(tmp->endTime, end);
  tmp->favorite = favorite;
  tmp->next = NULL;
  return tmp;
}
```

ฟังก์ชัน createEvent ทำหน้าที่สร้าง Struct event เปล่าเพื่อนำมาเก็บข้อมูลของ Task

createDay ()

```
//initialize Day node
Day *createDay(int day) {
  Day *tmp = (Day *)malloc(sizeof(Day));
  tmp->day = day;
  tmp->event = NULL;
  tmp->next = NULL;
  return tmp;
}
```

ฟังก์ชัน createDay ทำหน้าที่สร้าง Struct day เปล่าเพื่อนำมาเก็บวันที่ของ Task

createMonth ()

```
//initialize Month node
Month *createMonth(int month) {
  Month *tmp = (Month *)malloc(sizeof(Month));
  tmp->month = month;
  tmp->day = NULL;
  tmp->next = NULL;
  return tmp;
}
```

ฟังก์ชัน createMonth ทำหน้าที่สร้าง Struct month เปล่าเพื่อนำมาเก็บเดือนของ Task

createYear ()

```
//initialize Year node
Year *createYear(int year) {
  Year *tmp = (Year *)malloc(sizeof(Year));
  tmp->year = year;
  tmp->month = NULL;
  tmp->next = NULL;
  return tmp;
}
```

ฟังก์ชัน createYear ทำหน้าที่สร้าง Struct year เปล่าเพื่อนำมาเก็บปีของ Task

searchByYear ()

```
//search specific year
Year *searchByYear(Year *curYear, Year **prev, int year) {
  Year *current = curYear;
  while (current != NULL) {
    if (current->year == year) {
      return current;
    }
    //Keep track previousNode by prev
    *prev = current;
    current = current->next;
  }
  // Don't have any tasks on that year return NULL
  *prev = NULL;
  return current;
}
```

ฟังก์ชัน searchByYear ทำหน้าที่ return Struct year ที่ต้องการหา โดยรับปีเพื่อที่จะนำมาเทียบกับ Struct year ถ้าเทียบแล้วตรงกัน (หาเจอ) ฟังก์ชันจะ return Struct year ที่ตรงกับปีที่ต้องการ กรณีอื่นๆ ฟังก์ชันจะ return NULLซ

searchByYearFromOutdated ()

```
//search year in the stack if that year is already outdated
Year *searchByYearFromOutdated(Stack *outdated, int year) {

  int prevTop = outdated->top;
  //iterate over stack
  while (outdated->top != -1) {
    Year *headYear = peek(outdated);
    Year *current = headYear->next;
    if (current->year == year) {
      outdated->top = prevTop;
      //find that year and return current node
      return current;
    }
    pop(outdated);
  }
  outdated->top = prevTop;
  // Don't have any tasks on that Date return NULL
  return NULL;
}
```

ฟังก์ชัน searchByYearFromOutdated ทำหน้าที่ return Struct year ที่ต้องการหาจาก Stack โดยรับปีเพื่อที่จะนำมาเทียบกับ Struct year ใน Stack ถ้าเทียบแล้วตรงกัน (หาเจอ) ฟังก์ชันจะ return Struct year ที่ตรงกับปีที่ต้องการ กรณีอื่นๆ ฟังก์ชันจะ return NULL

searchByMonth ()

```
//search specific month
Month *searchByMonth(Month *curMonth, Month **prev, int
month) {
  Month *current = curMonth;
  while (current != NULL) {
    if (current->month == month) {
      return current;
    }
    *prev = current;
    current = current->next;
  }
  // Don't have any tasks on that month return NULL
  *prev = NULL;
  return current;
}
```

ฟังก์ชัน searchByMonth ทำหน้าที่ return Struct month ที่ต้องการหา โดยรับเดือนเพื่อที่จะนำมาเทียบกับ Struct month ถ้าเทียบแล้วตรงกัน (หาเจอ) ฟังก์ชันจะ return Struct month ที่ตรงกับเดือนที่ต้องการ กรณีอื่นๆ ฟังก์ชันจะ return NULL

searchByDay ()

```
//search specific day
Day *searchByDay(Day *curDay, Day **prev, int day) {
  Day *current = curDay;
  while (current != NULL) {
    if (current->day == day) {
      return current;
    }
    *prev = current;
    current = current->next;
  }
  // Don't have any tasks on that day return NULL
  *prev = NULL;
  return current;
}
```

ฟังก์ชัน searchByDay ทำหน้าที่ return Struct day ที่ต้องการหา โดยรับวันที่เพื่อที่จะนำมาเทียบกับ Struct day ถ้าเทียบแล้วตรงกัน (หาเจอ) ฟังก์ชันจะ return Struct day ที่ตรงกับวันที่ที่ต้องการ กรณีอื่นๆ ฟังก์ชันจะ return NULL

searchByDate ()

```
//search specific date
Day *searchByDate(Year *headYear, int year, int month, int day) {
  Year *prevYear = headYear;
  Year *tmpYear = searchByYear(headYear->next, &prevYear, year);
  //That year have some tasks
  if (tmpYear != NULL) {
    Month *prevMonth = tmpYear->month;
    Month *tmpMonth = searchByMonth(tmpYear->month->next,
&prevMonth, month);
    //That month have some tasks
    if (tmpMonth != NULL) {
      Day *prevDay = tmpMonth->day;
      Day *tmpDay = searchByDay(tmpMonth->day, &prevDay, day);
      //That day have some tasks and return pointer of day's node
      if (tmpDay != NULL) {
        return tmpDay;
      }
    }
  }
  // Don't have any tasks on that Date.
  return NULL;
}
```

ฟังก์ชัน searchByDate ทำหน้าที่ return Struct day ที่ต้องการหาด้วยการรับวันเดือนปี โดยจะเรียกใช้ฟังก์ชัน searchByYear, searchByMonth และ searchByDay เพื่อที่จะหา Struct day ที่ต้องการ ถ้าทั้ง 3 ฟังก์ชันนี้ไม่ return NULL ฟังก์ชัน searchByDate จะ return Struct day ที่ต้องการ กรณีอื่นๆ ฟังก์ชัน searchByDate จะ return NULL

searchByDateFromOutdated ()

```
Day *searchByDateFromOutdated(Stack *outdated, int year, int
month, int day) {
  int prevTop = outdated->top;
  //iterate data over stack
  while (outdated->top != -1) {
    Year *headYear = peek(outdated);
    Year *prevYear = headYear;
    Year *tmpYear = searchByYear(headYear->next, &prevYear, year);
    //That year have some tasks
    if (tmpYear != NULL) {
      Month *prevMonth = tmpYear->month;
      Month *tmpMonth = searchByMonth(tmpYear->month->next,
&prevMonth, month);
      //That month have some tasks
      if (tmpMonth != NULL) {
        Day *prevDay = tmpMonth->day;
        Day *tmpDay = searchByDay(tmpMonth->day, &prevDay, day);
        //That day have some tasks and return pointer of day's
node
        if (tmpDay != NULL) {
          outdated->top = prevTop;
          return tmpDay;
        }
      }
    }
    pop(outdated);
  }
  outdated->top = prevTop;
  // Don't have any tasks on that Date.
  return NULL;
}
```

ฟังก์ชัน searchByDateFromOutdated ทำหน้าที่ return Struct day ที่ต้องการหาจาก Stack ด้วยการรับวันเดือนปี โดยจะเรียกใช้ฟังก์ชัน searchByYear, searchByMonth และ searchByDay ถ้าทั้ง 3 ฟังก์ชันนี้ไม่ return NULL ฟังก์ชัน searchByDate จะ return Struct Date ที่ต้องการ กรณีอื่นๆ ฟังก์ชัน searchByDate จะ return NULL

displayCurrentTime ()

```
//show local time
void displayCurrentTime() {
  int yearTime, monthTime, dayTime;
  char nowTime[12];
  time_t current_time;
  struct tm *time_info;
  char date_string[20];

  // Get the current time
  current_time = time(NULL);

  // Convert the current time to local time
  time_info = localtime(&current_time);

  // Format the date string
  strftime(date_string, sizeof(date_string), "%d/%m/%y",
time_info);
  strftime(nowTime, sizeof(nowTime), "%H:%M", time_info);
  // Print the current date and local time
  sscanf(date_string, "%d/%d/%d", &dayTime, &monthTime,
&yearTime);
  printf("-----------------------\n");
  printf("Date : %d/%d/%d\n", dayTime, monthTime, yearTime);
  printf("Time : %s\n", nowTime);
}
```

ฟังก์ชัน displayCurrentTime ทำหน้าที่แสดงเวลาปัจจุบัน โดยจะเรียกใช้ฟังก์ชัน time เพื่อดึงเวลาปัจจุบันมาใช้ แล้วนำมาเปลี่ยนเป็น format ที่ต้องการแล้ว print ออกมา

displayFavorite ()

```
//show task that user mark favorite when create task or edit
favourite on that task
void displayFavorite(Year *headYear) {
  Year *curYear = headYear->next;
  if (curYear == NULL) {
    printf("No task\n");
  } else {
    while (curYear) {
      Month *curMonth = curYear->month->next;
      while (curMonth) {
        Day *curDay = curMonth->day->next;
        while (curDay) {
          Event *curEvent = curDay->event->next;
          while (curEvent) {
            if (curEvent->favorite != -1) {
              printf("Year: %d Month: %d  Day: %d Task: %s Time:
%s - %s "
                     "Favorite: %d\n",
                     curYear->year, curMonth->month, curDay->day,
                     curEvent->event, curEvent->startTime,
curEvent->endTime,
                     curEvent->favorite);
            }
            curEvent = curEvent->next;
          }
          curDay = curDay->next;
        }
        curMonth = curMonth->next;
      }
      curYear = curYear->next;
    }
  }
}
```

ฟังก์ชัน displayFavorite ทำหน้าที่แสดงผล Task มีความสำคัญและยังไม่เลยกำหนดเวลาทั้งหมด โดยจะไล่เช็กทีละ Task ว่ามีความสำคัญหรือไม่ ถ้ามี (favorite != -1) จะ print รายละเอียดของ Task ออกมา

displayByEvent ()

```
//Show all no-outdated event that have same task with user input
void displayByEvent(Year *headYear, char eventName[]) {
  printf("-----------------------\n");
  Year *curYear = headYear->next;
  if (curYear == NULL) {
    printf("No task\n");
  } else {
    while (curYear) {
      Month *curMonth = curYear->month->next;
      while (curMonth) {
        Day *curDay = curMonth->day->next;
        while (curDay) {
          Event *curEvent = curDay->event->next;
          int id = 1;
          while (curEvent) {
            //Check that task have the same name with user input or not
            if (!strcmp(eventName, curEvent->event)) {
              printf("Year: %d Month: %d  Day: %d\n", curYear->year,
                     curMonth->month, curDay->day);
              printf("   Task#%d %s - %s: %s\n", id,
curEvent->startTime,
                     curEvent->endTime, curEvent->event);
            }
            curEvent = curEvent->next;
            id++;
          }
          curDay = curDay->next;
        }
        curMonth = curMonth->next;
      }
      curYear = curYear->next;
    }
  }

  printf("\n");
}
```

ฟังก์ชัน displayByEvent ทำหน้าที่แสดงผล Task ที่ต้องการจากชื่อและยังไม่เลยกำหนดเวลาทั้งหมด โดยจะไล่เช็กทีละ Task ว่ามีชื่อตรงตามต้องการหรือเปล่า ถ้าตรง ฟังก์ชันจะ print รายละเอียดของ Task ออกมา

displayDayTask ()

```
//display all tasks that exist on day node
void displayDayTask(Day *curDay) {
  Event *curEvent = curDay->event->next;
  int id = 1;
  while (curEvent) {
    printf("   Task#%d %s - %s: %s\n", id,
curEvent->startTime,
           curEvent->endTime, curEvent->event);
    curEvent = curEvent->next;
    id++;
  }
}
```

ฟังก์ชัน displayDayTask ทำหน้าที่แสดงผลเวลาและชื่อของ Task ในวันนั้นๆ ทั้งหมด โดยรับ Struct day เพื่อที่จะ print ข้อมูลใน Struct event ออกมา

displayByDate ()

```
void displayByDate(Year *headYear, int year, int month,
int day) {
  Day *current = searchByDate(headYear, year, month,
day);
  if (current) {
    printf("Year: %d Month: %d Day: %d\n", year, month,
day);
    displayDayTask(current);
  } else {
    printf("No task\n");
  }
}
```

ฟังก์ชัน displayByDate ทำหน้าที่แสดงผลข้อมูลของ Task ทั้งหมดในวันเดือนปีนั้นๆ โดยรับวันเดือนปี แล้วนำมาเรียกใช้ฟังก์ชัน searchByDate เพื่อที่จะเช็กว่ามี Task ในวันนั้นๆ หรือเปล่า ถ้ามีฟังก์ชันจะ print วันเดือนปีและเรียกใช้ฟังก์ชัน displayDayTask เพื่อ print เวลาและชื่อของ Task ออกมา แต่ถ้าไม่มี ฟังก์ชันจะ print “No task”

displayByDay ()

```
void displayByDay(Year *headYear, int day) {
  Year *curYear = headYear->next;
  while (curYear) {
    Month *headMonth = curYear->month;
    Month *curMonth = headMonth->next;
    while (curMonth) {
      Day *headDay = curMonth->day;
      Day *prevDay = headDay;
      Day *curDay = searchByDay(headDay->next, &prevDay,
day);
      if (curDay) {
        printf("Year: %d Month: %d Day: %d\n", curYear->year,
curMonth->month,
               curDay->day);
        displayDayTask(curDay);
      }
      curMonth = curMonth->next;
    }

    curYear = curYear->next;
  }
}
```

ฟังก์ชัน displayByDay ทำหน้าที่แสดงผลข้อมูลของ Task ทั้งหมดในวันที่นั้นๆ โดยรับวันที่ แล้วนำมาเรียกใช้ฟังก์ชัน searchByDay เพื่อที่จะเช็กว่ามี Task ในวันที่นั้นๆ หรือเปล่า ถ้ามี ฟังก์ชันจะ print วันเดือนปี และเรียกใช้ฟังก์ชัน displayDayTask เพื่อ print เวลาและชื่อของ Task ออกมา

displayCurrentDateTask ()

```
//display all task in the current day
void displayCurrentDateTask(Year *headYear) {
  // Get RealTimeDate
  int yearTime, monthTime, dayTime;
  char nowTime[12];
  time_t current_time;
  struct tm *time_info;
  char date_string[20];

  // Get the current time
  current_time = time(NULL);

  // Convert the current time to local time
  time_info = localtime(&current_time);

  // Format the date string
  strftime(date_string, sizeof(date_string), "%d/%m/%y",
time_info);
  strftime(nowTime, sizeof(nowTime), "%H:%M", time_info);
  // Print the current date and local time
  sscanf(date_string, "%d/%d/%d", &dayTime, &monthTime,
&yearTime);

  // Daily notification
  printf("Today tasks: \n");
  displayByDate(headYear, yearTime, monthTime, dayTime);
}
```

ฟังก์ชัน displayCurrentDateTask ทำหน้าที่แสดงผล Task ทั้งหมดที่มีกำหนดการตรงกับวันเดือนปีปัจจุบัน โดยจะเรียกใช้ฟังก์ชัน time เพื่อดึงเวลาปัจจุบันมาใช้ แล้วนำมาเปลี่ยนเป็นวันเดือนปี จากนั้นก็เรียกใช้ฟังก์ชัน displayByDate เพื่อที่จะ print ข้อมูลทั้งหมดของ Task ในวันเดือนปีปัจจุบัน

displayByMonth ()

```
//display all task that exist on the specific month
void displayByMonth(Year *headYear, int month) {
  Year *curYear = headYear->next;
  while (curYear) {
    Month *headMonth = curYear->month;
    Month *prevMonth = headMonth;
    Month *curMonth = searchByMonth(headMonth->next,
&prevMonth, month);
    if (curMonth) {
      Day *headDay = curMonth->day;
      Day *curDay = headDay->next;
      while (curDay) {
        printf("Year: %d Month: %d Day: %d\n",
curYear->year, month,
               curDay->day);
        displayDayTask(curDay);
        curDay = curDay->next;
      }
    } else {
      printf("Year: %d - Don't have any tasks in that
year\n", curYear->year);
    }
    curYear = curYear->next;
  }
}
```

ฟังก์ชัน displayByMonth ทำหน้าที่แสดงผลข้อมูลของ Task ทั้งหมดในเดือนนั้นๆ โดยรับเดือน แล้วนำมาเรียกใช้ฟังก์ชัน searchByMonth เพื่อที่จะเช็กว่ามี Task ในเดือนนั้นๆ หรือเปล่า ถ้ามีฟังก์ชันจะ print วันเดือนปีและเรียกใช้ฟังก์ชัน displayDayTask เพื่อ print เวลาและชื่อของ Task ออกมา

displayByYear ()

```
//display all task that exist on the specific year
void displayByYear(Year *headYear, int year) {
  Year *prevYear = headYear;
  Year *curYear = searchByYear(headYear, &prevYear, year);

  //Check if we have any tasks on that year
  if (curYear) {
    printf("Year: %d", year);
    Month *headMonth = curYear->month;
    Month *curMonth = headMonth->next;
    while (curMonth) {
      printf("\nMonth %d\n", curMonth->month);
      Day *headDay = curMonth->day;
      Day *curDay = headDay->next;
      while (curDay) {
        printf("Day %d:\n", curDay->day);
        displayDayTask(curDay);
        curDay = curDay->next;
      }
      curMonth = curMonth->next;
    }
  } else {
    printf("Don't have any tasks in that year\n");
  }
}
```

ฟังก์ชัน displayByYear ทำหน้าที่แสดงผลข้อมูลของ Task ทั้งหมดในปีนั้นๆ โดยรับปี แล้วนำมาเรียกใช้ฟังก์ชัน searchByYear เพื่อที่จะเช็กว่ามี Task ในปีนั้นๆ หรือเปล่า ถ้ามี ฟังก์ชัน displayByYear จะ print วันเดือนปีและเรียกใช้ฟังก์ชัน displayDayTask เพื่อ print เวลาและชื่อของ Task ออกมา แต่ถ้าไม่มี ฟังก์ชัน displayByYear จะ print "Don't have any tasks in that year"

displayOutdated ()

```
//display the outdated task (Order by the nearest current time)
void displayOutdated(Stack *outdated) {
  int prevTop = outdated->top;
  // iterate over stack
  while (outdated->top != -1) {
    Year *outdatedHeadYear = peek(outdated);
    // display all outdated task in that year
    displayByYear(outdatedHeadYear, outdatedHeadYear->next->year);
    pop(outdated);
  }
  outdated->top = prevTop;
}
```

ฟังก์ชัน displayOutdated ทำหน้าที่แสดงผล Task ที่เลยกำหนดเวลาไปแล้วทั้งหมด โดยนำ Stack ของ Task ที่เลยกำหนดเวลาไปแล้วมาเรียกใช้ฟังก์ชัน peek เพื่อที่จะนำปีที่ได้มาเรียกใช้ฟังก์ชัน displayByYear โดยฟังก์ชันนี้จะ print ข้อมูลของ Task ทั้งหมดในปีนั้นๆ แล้วเรียกใช้ฟังก์ชัน pop ทำแบบนี้ไปเรื่อยๆ จนกว่า top จะเท่ากับ -1

displayAll ()

```
//display all tasks
void displayAll(Year *headYear, int opr, int *allTask) {
  printf("-----------------------\n");
  Year *curYear = headYear->next;
  int id = 1;
  if (curYear == NULL) {
    printf("No task\n");
  } else {
    // Use to edit event
    if (opr == 1) {
      id = 1;
    }
    while (curYear) {
      printf("Year: %d\n", curYear->year);
      Month *curMonth = curYear->month->next;
      while (curMonth) {
        printf("  Month: %d\n", curMonth->month);
        Day *curDay = curMonth->day->next;
        while (curDay) {
```

```
          printf("     Day: %d\n", curDay->day);
          Event *curEvent = curDay->event->next;
          // Display normally
          if (opr == 2) {
            id = 1;
          }
          while (curEvent) {
            printf("       Task#%d %s - %s: %s\n", id,
curEvent->startTime,
                   curEvent->endTime, curEvent->event);
            curEvent = curEvent->next;
            id++;
            *allTask++;
          }
          curDay = curDay->next;
        }

        curMonth = curMonth->next;
      }

      curYear = curYear->next;
    }
  }

  printf("\n");
}
```

ฟังก์ชั่น displayAll ทำหน้าที่แสดงผล Task ที่ยังไม่เลยกำหนดเวลาทั้งหมด โดยแสดงผล Task ตามวันและเวลาที่ใกล้กับเวลาปัจจุบันมากที่สุดก่อน

deleteFirstEvent ()

```
//delete first event (like dequeue on queue)
void deleteFirstEvent(Year* headYear, int number, int year, int month,
int day, int* status){

    Day* curDay = searchByDate(headYear,year,month,day);
    //Of course, curDay won't be null

    while(number--){
        Event* headEvent = curDay->event;
        Event* prevEvent = headEvent;
        Event* curEvent  = headEvent->next;
        prevEvent->next  = curEvent->next;
        free(curEvent);

        //If don't have any task left then free that day
        if(headEvent->next == NULL){
            //free head of event in that day
            free(headEvent);

            //prev will keep track on node to delete that node;
            Year* prevYear = headYear;
            Year* tmpYear = searchByYear(headYear->next,&prevYear,year);

            Month* headMonth = tmpYear->month;
            Month* prevMonth = headMonth;
            Month* tmpMonth =
searchByMonth(tmpYear->month->next,&prevMonth,month);

            Day* headDay = tmpMonth->day;
            Day* prevDay = headDay;
            Day* tmpDay = searchByDay(tmpMonth->day->next,&prevDay,day);


            prevDay->next = tmpDay->next;
            free(tmpDay);
            //If don't have any task left on any days in that month then
free that month
            if(headDay->next == NULL){
                free(headDay);

                prevMonth->next = tmpMonth->next;
                free(tmpMonth);

                //If don't have any task left on any months in that year
```

```
then free that year
                if(headMonth->next == NULL){
                    free(headMonth);

                    prevYear->next = tmpYear->next;
                    free(tmpYear);
                    //free year already and turn status to 1
                    *status = 1;

                }
            }

        }
    }
}
```

ฟังก์ชั่น deleteFirstEvent ทำหน้าที่ลบ Task แรกสุด โดยรับจำนวนที่ต้องการลบและวันเดือนปีของ Task แรกสุด แล้วนำมาเรียกใช้ฟังก์ชัน searchByDate เพื่อที่นำ Struct day มา free Task แรกสุด ถ้าในวันเดือนปีไหนๆ ไม่มี Task ในวันเดือนปีนั้นๆ แล้ว ให้ free วันเดือนปีนั้นๆ ฟังก์ชันจะทำซ้ำเรื่อยๆ จนกว่าจะถึงจำนวนที่ต้องการลบ

deleteEvent ()

```
// delete the task that on the selected date
void deleteEvent(Year *headYear, Stack *outdated, int year, int month,
                 int day) {
  char buffer[120];

  Day *curDay = searchByDate(headYear, year, month, day);
  Day *curOutdatedDay = searchByDateFromOutdated(outdated, year, month,
day);
  int id = 1;
  // display the task that doesn't outdated on that day
  if (curDay) {
    printf("Year: %d Month: %d Day: %d\n", year, month, day);
    Event *curEvent = curDay->event->next;
    while (curEvent) {
      printf("   Task#%d %s - %s: %s\n", id, curEvent->startTime,
             curEvent->endTime, curEvent->event);
      curEvent = curEvent->next;
      id++;
    }
```

```
}
// display the task that already outdated on that day
if (curOutdatedDay) {
  printf("--------------OUTDATED!!!--------------\n");
  printf("Year: %d Month: %d Day: %d\n", year, month, day);
  Event *curEvent = curOutdatedDay->event->next;
  while (curEvent) {
    printf("   Task#%d %s - %s: %s\n", id, curEvent->startTime,
           curEvent->endTime, curEvent->event);
    curEvent = curEvent->next;
    id++;
  }
}
//No task on that day
if (!curDay && !curOutdatedDay) {
  printf("Sorry you don't have any task in that date\n");
  return;
}

// let the user select task to delete
int task = 0;
printf("Select Task that you want to delete by number: ");
fgets(buffer, 120, stdin);
enterRemover(buffer);
sscanf(buffer, "%d", &task);
while (task < 1 || task > id) {
  printf("Please select in the range (1-%d): \n", id);
  fgets(buffer, 120, stdin);
  enterRemover(buffer);
  sscanf(buffer, "%d", &task);
}
id = 1;
//Delete part
if (curDay) {
  Event *curEvent = curDay->event->next;
  Event *prevEvent = curDay->event;
  while (curEvent) {
    if (id == task) {
      printf("Remember delete can't be undo\n");
      int choice = 0;
      while (choice != 1 && choice != 2) {
        printf("Choice 1: Yes (Delete it)\n");
        printf("Choice 2: No (Cancel it)\n");
        printf("Select choice : ");
        fgets(buffer, 120, stdin);
        enterRemover(buffer);
```

```
          sscanf(buffer, "%d", &choice);
        }
        if (choice == 1) {

          Event *headEvent = curDay->event;
          prevEvent->next = curEvent->next;
          free(curEvent);

          // If don't have any task left then free that day
          if (headEvent->next == NULL) {
            // free head of event in that day
            free(headEvent);

            // prev will keep track on node to delete that node;

            Year *prevYear = headYear;
            Year *tmpYear = searchByYear(headYear->next, &prevYear,
year);

            Month *headMonth = tmpYear->month;
            Month *prevMonth = headMonth;
            Month *tmpMonth =
                searchByMonth(tmpYear->month->next, &prevMonth, month);

            Day *headDay = tmpMonth->day;
            Day *prevDay = headDay;
            Day *tmpDay = searchByDay(tmpMonth->day->next, &prevDay,
day);

            prevDay->next = tmpDay->next;
            free(tmpDay);
            // If don't have any task left on any days in that month
then free
            // that month;
            if (headDay->next == NULL) {
              free(headDay);

              prevMonth->next = tmpMonth->next;
              free(tmpMonth);

              // If don't have any task left on any months in that year
then
              // free that year;
              if (headMonth->next == NULL) {
                free(headMonth);
```

```
                prevYear->next = tmpYear->next;
                free(tmpYear);
              }
            }
          }
          printf("Deletion Success!!!\n");
          return;
        } else {
          printf("Cancel Success!!!\n");
          return;
        }
      }
      id++;
      prevEvent = curEvent;
      curEvent = curEvent->next;
    }
  }
  // printf("******Outdatd Par******\n");
  //  all event in outdatedData
  if (curOutdatedDay) {
    Event *curEvent = curOutdatedDay->event->next;
    Event *prevEvent = curOutdatedDay->event;
    while (curEvent) {
      printf("Here %d %d\n", id, task);
      if (id == task) {
        printf("Remember delete can't be undo\n");
        int choice = 0;
        while (choice != 1 && choice != 2) {
          printf("Choice 1: Yes (Delete it)\n");
          printf("Choice 2: No (Cancel it)\n");
          printf("Select choice : ");
          fgets(buffer, 120, stdin);
          enterRemover(buffer);
          sscanf(buffer, "%d", &choice);
        }
        if (choice == 1) {

          Event *headEvent = curOutdatedDay->event;
          prevEvent->next = curEvent->next;
          free(curEvent);

          // If don't have any task left then free that day
          if (headEvent->next == NULL) {
            // free head of event in that day
            free(headEvent);
```

```
            // prev will keep track on node to delete that node;
            Year *outdatedHeadYear = searchByYearFromOutdated(outdated,
year);
            Year *prevYear = outdatedHeadYear;
            Year *tmpYear = outdatedHeadYear->next;

            Month *outdatedHeadMonth = tmpYear->month;
            Month *prevMonth = outdatedHeadMonth;
            Month *tmpMonth =
                searchByMonth(tmpYear->month->next, &prevMonth, month);

            Day *outdatedHeadDay = tmpMonth->day;
            Day *prevDay = outdatedHeadDay;
            Day *tmpDay = searchByDay(tmpMonth->day->next, &prevDay,
day);

            prevDay->next = tmpDay->next;
            free(tmpDay);
            // If don't have any task left on any days in that month
then free
            // that month;
            if (outdatedHeadDay->next == NULL) {
              free(outdatedHeadDay);

              prevMonth->next = tmpMonth->next;
              free(tmpMonth);

              // If don't have any task left on any months in that year
then
              // free that year;
              if (outdatedHeadMonth->next == NULL) {
                free(outdatedHeadMonth);

                prevYear->next = tmpYear->next;
                free(tmpYear);

                // shiff array

                outdatedHeadYear->next = NULL;
                int prevTop = outdated->top;
                int flag = 0;
                //  printf("Head %s\n",outdatedHeadYear->next);
                for (int i = 0; i <= prevTop; i++) {
                  if (flag == 1) {

                    outdated->arr[i - 1] = outdated->arr[i];
```

```
                }
                if (outdated->arr[i]->next == NULL) {

                  flag = 1;
                }
              }
              if (flag == 1) {
                outdated->top--;
              }
            }
          }
        }
        printf("Deletion Outdated Success!!!\n");
        return;
      } else {
        printf("Cancel Success!!!\n");
        return;
      }
    }
    id++;
    prevEvent = curEvent;
    curEvent = curEvent->next;
  }
 }
}
```

ฟังก์ชั่น deleteEvent ทำหน้าที่เลือกลบ Task ในวันเดือนปีที่ต้องการ โดยรับวันเดือนปีแล้วนำมาเรียกใช้ฟังก์ชัน searchByDate และ searchByDateFromOutdated เพื่อที่จะเช็กว่าวันเดือนปีนี้มี Task รึเปล่า ถ้ามีก็จะ print Task ทั้งหมดในวันเดือนปีนั้นๆ เพื่อให้ User เลือกว่าจะลบ Task ไหน หลังจากนั้นก็จะให้ User ยืนยันการลบ Task อีกครั้ง

insert ()

```
// create the task and insert into tree
void insert(Year *headYear, int year, int month, int day, char start[],
            char end[], char event[], int favorite) {

  // sort year (increasing order)
  Year *prevYear = headYear;
  Year *curYear = headYear->next;
  Year *tmpYear = searchByYear(curYear, &prevYear, year);

  //if tmpyear is NULL initialize header node for that year
  if (tmpYear == NULL) {
    Year *newYear = createYear(year);
    tmpYear = newYear;
    if (curYear == NULL) {
      headYear->next = newYear;
    } else if (curYear->year > year) {
      newYear->next = curYear;
      headYear->next = newYear;

    } else {
      while (curYear->next && curYear->next->year < year) {
        curYear = curYear->next;
      }
      newYear->next = curYear->next;
      curYear->next = newYear;
    }
    tmpYear->month = createMonth(-1);
  }

  // sort month
  Month *headMonth = tmpYear->month;
  Month *curMonth = headMonth->next;
  Month *prevMonth = headMonth;
  Month *tmpMonth = searchByMonth(curMonth, &prevMonth, month);

  //if tmpmonth is NULL initialize header node for that month
  if (tmpMonth == NULL) {
    Month *newMonth = createMonth(month);
    tmpMonth = newMonth;
    if (curMonth == NULL) {
      headMonth->next = newMonth;
    } else if (curMonth->month > month) {
      newMonth->next = curMonth;
      headMonth->next = newMonth;
```

```
    } else {
      while (curMonth->next && curMonth->next->month < month) {
        curMonth = curMonth->next;
      }
      newMonth->next = curMonth->next;
      curMonth->next = newMonth;
    }
    tmpMonth->day = createDay(-1);
  }

  // sort Day
  Day *headDay = tmpMonth->day;
  Day *curDay = headDay->next;
  Day *prevDay = headDay;
  Day *tmpDay = searchByDay(tmpMonth->day, &prevDay, day);

  //if tmpday is NULL initialize header node for that day
  if (tmpDay == NULL) {
    Day *newDay = createDay(day);
    tmpDay = newDay;
    if (curDay == NULL) {
      headDay->next = newDay;
    } else if (curDay->day > day) {
      newDay->next = curDay;
      headDay->next = newDay;

    } else {
      while (curDay->next && curDay->next->day < day) {
        curDay = curDay->next;
      }
      newDay->next = curDay->next;
      curDay->next = newDay;
    }
    tmpDay->event = createEvent("HEAD", "0", "0", -1);
  }

  // sort Star & End and if star and time is equal to another tasks it
will sort by favorite (FCFS)
  Event *headEvent = tmpDay->event;
  Event *curEvent = headEvent->next;
  Event *newEvent = createEvent(event, start, end, favorite);

  char curEventTime[12];
  char newEventTime[12];
```

```
  strcpy(newEventTime, strcat(start, end));

  if (curEvent != NULL) {
    strcpy(curEventTime, curEvent->startTime);
    strcat(curEventTime, curEvent->endTime);
    //
strcpy(curEventTime,strcat(curEvent->startTime,curEvent->endTime));
  }

  if (curEvent == NULL) {
    headEvent->next = newEvent;
  }

  else if (strcmp(newEventTime, curEventTime) < 0) {
    newEvent->next = curEvent;
    headEvent->next = newEvent;
  }

  else {

    if (curEvent->next != NULL) {
      strcpy(curEventTime, curEvent->next->startTime);
      strcat(curEventTime, curEvent->next->endTime);
      while (curEvent->next && strcmp(newEventTime, curEventTime) > 0) {
        curEvent = curEvent->next;
        if (curEvent->next) {
          strcpy(curEventTime, curEvent->next->startTime);
          strcat(curEventTime, curEvent->next->endTime);
        }
      }
      if (strcmp(newEventTime, curEventTime) == 0) {
        if (favorite > curEvent->next->favorite) {
          newEvent->next = curEvent->next;

          curEvent->next = newEvent;

        } else {
          while (curEvent->next && (strcmp(newEventTime, curEventTime)
== 0) &&
                 favorite <= curEvent->next->favorite) {
            curEvent = curEvent->next;

            if (curEvent->next) {
              strcpy(curEventTime, curEvent->next->startTime);
              strcat(curEventTime, curEvent->next->endTime);
            }
```

```
          }
          newEvent->next = curEvent->next;
          curEvent->next = newEvent;
        }
      }
    } else {
      newEvent->next = curEvent->next;
      curEvent->next = newEvent;
    }
  }
}
```

ฟังก์ชัน insert ทำหน้าที่สร้างและแทรก Task ใหม่ลงไป Tree โดยรับข้อมูลของ Task มาแล้วเรียกใช้ฟังก์ชัน searchByYear เช็กว่า Task ใหม่นี้มี Struct year หรือยัง ถ้ายังไม่มี ให้เรียกใช้ฟังก์ชัน createYear เพื่อสร้าง Struct year ใหม่แล้วไล่เช็กตั้งแต่ head Struct year เพื่อที่จะหาตำแหน่งที่ Struct year ใหม่ควรอยู่โดยเรียงจากน้อยไปมาก จากนั้นเช็กเดือน, วัน, เวลา และความสำคัญของ Task ว่าต้องสร้าง Struct ใหม่หรือต้องหาตำแหน่งที่ควรอยู่ใหม่

userInput()

```c
// get all input form user
void userInput(int *year, int *month, int *day, char *start, char *end,
               char *event, int *favorite, int type) {

  // type 1 getAll (displayAll)
  // type 2 getDate (deleteEvent,editEvent(SelectPart))
  // type 3 forSearch (What to search)
  // type 4 forEdit (What to Edit)

  // get date
  char buffer[120];
  if (type == 1 || type == 2) {

    printf("input dd/mm/yy (01/01/23): ");
    fgets(buffer, 120, stdin);
    enterRemover(buffer);
    while (!validateDate(buffer)) {
      printf("wrong format please try again\n");
      printf("input dd/mm/yy (01/01/23): ");
      fgets(buffer, 120, stdin);
      enterRemover(buffer);
    }
    sscanf(buffer, "%d/%d/%d;", day, month, year);

  }
  if (type == 3 || type == 4) {
    printf("input dd/mm/yy (01/01/23): ");
    fgets(buffer, 120, stdin);
    enterRemover(buffer);
    while (!findDateBlank(buffer, day, month, year)) {
      printf("wrong format please try again\n");
      printf("input dd/mm/yy (01/01/23): ");
      fgets(buffer, 120, stdin);
    }
  }

  // get start and end time
  if (type == 4) {
    printf("input time to start and end in format (xx:yy)\n");
    printf("Time to start: ");
    fgets(start, 12, stdin);
    enterRemover(start);
    while (!findTimeBlank(start)) {
      printf("wrong format please try again\n");
```

```
      printf("input time in format (xx:yy)\n");
      printf("Time to start: ");
      fgets(start, 12, stdin);
      enterRemover(start);
    }

    printf("Time to end: ");
    fgets(end, 12, stdin);
    enterRemover(end);
    while (!findTimeBlank(end)) {
      printf("wrong format please try again\n");
      printf("input time in format (xx:yy)\n");
      printf("Time to end: ");
      fgets(end, 12, stdin);
      enterRemover(end);
    }
  }

  if (type == 1) {
    printf("input time to start and end in format (xx:yy)\n");
    printf("Time to start: ");
    fgets(start, 12, stdin);
    enterRemover(start);
    while (!validateTime(start)) {
      printf("wrong format please try again\n");
      printf("input time in format (xx:yy)\n");
      printf("Time to start: ");
      fgets(start, 12, stdin);
      enterRemover(start);
    }

    printf("Time to end: ");
    fgets(end, 12, stdin);
    enterRemover(end);
    while (!validateTime(end)) {
      printf("wrong format please try again\n");
      printf("input time in format (xx:yy)\n");
      printf("Time to end: ");
      fgets(end, 12, stdin);
      enterRemover(end);
    }

    while (!(strcmp(start, end) < 0)) {
      printf("uncorrect time period please time again\n");
      printf("input time to start and end in format (xx:yy)\n");
      printf("Time to start: ");
```

```
      fgets(start, 12, stdin);
      enterRemover(start);
      while (!validateTime(start)) {
        printf("wrong format please try again\n");
        printf("input time in format (xx:yy)\n");
        printf("Time to start: ");
        fgets(start, 12, stdin);
        enterRemover(start);
      }

      printf("Time to end: ");
      fgets(end, 12, stdin);
      enterRemover(end);
      while (!validateTime(end)) {
        printf("wrong format please try again\n");
        printf("input time in format (xx:yy)\n");
        printf("Time to end: ");
        fgets(end, 12, stdin);
        enterRemover(end);
      }
    }
  }

  //Get task name
  if (type == 1 || type == 3 || type == 4) {
    printf("Name your Task: ");
    fgets(event, 3000, stdin);
    trim(event);
    enterRemover(event);
  }

  // Get the favorite value
  if (type == 1) {
    printf("Please rate how important this task is\n");
    printf("insert \"-1\" if you think that not important\n");
    printf("input( any number that more than or equal zero ) : ");
    fgets(buffer, 120, stdin);
    enterRemover(buffer);
    sscanf(buffer, "%d", favorite);
    while (*favorite < -1) {
      printf("wrong format please try again\n");
      printf("Please rate how important this task is \n");
      printf("insert \"-1\" if you think that not important\n");
      printf("input( any number that more than zero ) : ");
      fgets(buffer, 120, stdin);
      enterRemover(buffer);
```

```
      sscanf(buffer, "%d", favorite);
    }
  }

  if (type == 4) {
    printf("Please rate how important this task is\n");
    printf("input( any number that more than or equal zero ) : ");
    fgets(buffer, 120, stdin);
    enterRemover(buffer);
    while (!findFavBlank(buffer, favorite)) {
      printf("%d\n", *favorite);
      printf("wrong format please try again\n");
      printf("Please rate how important this task is \n");
      printf("input( any number that more than zero ) : ");
      fgets(buffer, 120, stdin);
      enterRemover(buffer);
    }
  }

}
```

ฟังก์ชัน userInput ทำหน้าที่รับข้อมูลจาก User โดยจะแบ่งการรับข้อมูลออกเป็น 4 แบบ และทุกการรับข้อมูลจะมีการเช็กว่าข้อมูลตรงตามรูปแบบหรือไม่

1. รับข้อมูลวัน เดือน ปี เวลา ชื่อ และความสำคัญของ Task
2. รับข้อมูลวัน เดือน และปีของ Task
3. รับข้อมูลวัน เดือน ปี และชื่อของ Task ที่สามารถเป็น “--” ได้
4. รับข้อมูลวัน เดือน ปี เวลา ชื่อ และความสำคัญของ Task ที่สามารถเป็น “--” ได้

edit ()

```
// Edit task by delete the old one and insert the edit one
void edit(Year *headYear, int year, int month, int day) {
  char buffer[120];
  Day *curDay = searchByDate(headYear, year, month, day);
  int id = 1;
  // Show the task that can be selected on that day
  if (curDay) {
    printf("Year: %d Month: %d Day: %d\n", year, month, day);
    Event *curEvent = curDay->event->next;
    while (curEvent) {
      printf("   Task#%d %s - %s: %s\n", id, curEvent->startTime,
             curEvent->endTime, curEvent->event);
      curEvent = curEvent->next;
      id++;
    }
  } else {
    printf("Sorry you don't have any task in that date\n");
    return;
  }
  // Select the task on that day
  int task = 0;
  printf("Select Task that you want to edit by number: ");
  fgets(buffer, 120, stdin);
  enterRemover(buffer);
  sscanf(buffer, "%d", &task);
  while (task < 1 || task > id) {
    printf("Please select in the range (1-%d): \n", id);
    fgets(buffer, 120, stdin);
    enterRemover(buffer);
    sscanf(buffer, "%d", &task);
  }

  int editYear = -1, editMonth = -1, editDay = -1, editFavorite = -1;
  char editStart[12], editEnd[12], editEvent[3000];

  Event *curEvent = curDay->event->next;
  Event *prevEvent = curDay->event;
  id = 1;
  while (curEvent) {
    if (id == task) {

      // Get What to edit
      printf("Please insert all data you want to edit\n");
      printf("(leave \"--\" if don't want to edit in that fields we will
```

```
use "
             "the old ones)\n");
      userInput(&editYear, &editMonth, &editDay, editStart, editEnd,
editEvent,
                &editFavorite, 4);

      //if that field has "--" keep old data
      if (editYear == -1)
        editYear = year;
      if (editMonth == -1)
        editMonth = month;
      if (editDay == -1)
        editDay = day;
      if (editFavorite == -2)
        editFavorite = curEvent->favorite;
      if (!strcmp(editStart, "--"))
        strcpy(editStart, curEvent->startTime);
      if (!strcmp(editEnd, "--"))
        strcpy(editEnd, curEvent->endTime);
      if (!strcmp(editEvent, "--"))
        strcpy(editEvent, curEvent->event);

      // check if user input was right
      while (!validateDateNum(editYear, editMonth, editDay) ||
             !(strcmp(editStart, editEnd) < 0)) {
        printf("There is something wrong when using the new data and the
old "
               "data together please try again\n");
        userInput(&editYear, &editMonth, &editDay, editStart, editEnd,
                  editEvent, &editFavorite, 4);
        if (editYear == -1)
          editYear = year;
        if (editMonth == -1)
          editMonth = month;
        if (editDay == -1)
          editDay = day;
        if (editFavorite == -2)
          editFavorite = curEvent->favorite;
        if (!strcmp(editStart, "--"))
          strcpy(editStart, curEvent->startTime);
        if (!strcmp(editEnd, "--"))
          strcpy(editEnd, curEvent->endTime);
        if (!strcmp(editEvent, "--"))
          strcpy(editEvent, curEvent->event);
      }
```

```
      // Delete
      Event *headEvent = curDay->event;
      prevEvent->next = curEvent->next;
      free(curEvent);

      // If don't have any task left then free that day
      if (headEvent->next == NULL) {
        // free head of event in that day
        free(headEvent);

        // prev will keep track on node to delete that node;
        Year *prevYear = headYear;
        Year *tmpYear = searchByYear(headYear->next, &prevYear, year);

        Month *headMonth = tmpYear->month;
        Month *prevMonth = headMonth;
        Month *tmpMonth =
            searchByMonth(tmpYear->month->next, &prevMonth, month);

        Day *headDay = tmpMonth->day;
        Day *prevDay = headDay;
        Day *tmpDay = searchByDay(tmpMonth->day->next, &prevDay, day);

        prevDay->next = tmpDay->next;
        free(tmpDay);
        // If don't have any task left on any days in that month then
free that
        // month;
        if (headDay->next == NULL) {
          free(headDay);

          prevMonth->next = tmpMonth->next;
          free(tmpMonth);

          // If don't have any task left on any months in that year then
free
          // that year;
          if (headMonth->next == NULL) {
            free(headMonth);

            prevYear->next = tmpYear->next;
            free(tmpYear);
          }
        }
      }
    }
```

```
    id++;
    prevEvent = curEvent;
    curEvent = curEvent->next;
  }

  // inserted edited data
  insert(headYear, editYear, editMonth, editDay, editStart, editEnd,
editEvent, editFavorite);
}
```

ฟังก์ชัน edit ทำหน้าที่แก้ไข Task ที่ต้องการ โดยการลบ Task ที่เลือกแล้วสร้าง Task ใหม่ขึ้นมาแทน เริ่มจากการเรียกใช้ฟังก์ชัน searchByDate เพื่อที่จะเช็กว่ามี Task ในวันเดือนปีนั้นๆ หรือไม่ จากนั้นก็แสดงผล Task ทั้งหมดในวันเดือนปีนั้นเพื่อให้ User เลือก Task ที่จะแก้ไข และให้ User แก้ไขวัน เดือน ปี เวลา และความสำคัญของ Task แล้วตรวจสอบว่าข้อมูลที่แก้ไขยังตรงตามรูปแบบอยู่หรือไม่ ต่อด้วยการลบ Task นั้นๆ ด้วยการ free หลังจากการ free ถ้ามี Struct event day month หรือ year ไหนที่เป็น NULL ให้ free ด้วยเช่นกัน จบด้วยการนำข้อมูลที่ User ได้ทำการแก้ไขมาเรียกใช้ฟังก์ชัน insert

stackHelper ()

```
///help to insert data in the right location inside stack
void stackHelper(Stack* outdated, int year, int month, int day,char
start[],char end[], char event[], int favorite){
  Year *tmpYear = searchByYearFromOutdated(outdated,year);
  if (tmpYear == NULL) {
    tmpYear = createYear(-1);
    push(outdated,tmpYear);
    Year *newYear = createYear(year);
    tmpYear->next = newYear;
    tmpYear = newYear;
    tmpYear->month = createMonth(-1);
  }
  // sort month
  Month *headMonth = tmpYear->month;
  Month *curMonth = headMonth->next;

  Month *prevMonth = headMonth;

  Month *tmpMonth = searchByMonth(curMonth,&prevMonth, month);

  if (tmpMonth == NULL) {
    Month *newMonth = createMonth(month);
    tmpMonth = newMonth;
    if (curMonth == NULL) {
      headMonth->next = newMonth;
    } else if (curMonth->month > month) {
      newMonth->next = curMonth;
      headMonth->next = newMonth;

    } else {
      while (curMonth->next && curMonth->next->month < month) {
        curMonth = curMonth->next;
      }
      newMonth->next = curMonth->next;
      curMonth->next = newMonth;
    }
    tmpMonth->day = createDay(-1);
  }
  // sort Day
  Day *headDay = tmpMonth->day;
  Day *curDay = headDay->next;
  Day *prevDay = headDay;
  Day *tmpDay = searchByDay(tmpMonth->day, &prevDay, day);
  if (tmpDay == NULL) {
```

```
    Day *newDay = createDay(day);
    tmpDay = newDay;
    if (curDay == NULL) {
      headDay->next = newDay;
    } else if (curDay->day > day) {
      newDay->next = curDay;
      headDay->next = newDay;

    } else {
      while (curDay->next && curDay->next->day < day) {
        curDay = curDay->next;
      }
      newDay->next = curDay->next;
      curDay->next = newDay;
    }
    tmpDay->event = createEvent("HEAD", "0", "0",-1);
  }

  //sort Star & End and if star and time is equal to another tasks it
will sort by favorite (FCFS)
  Event *headEvent = tmpDay->event;
  Event *curEvent = headEvent->next;
  Event *newEvent = createEvent(event, start, end,favorite);

  char curEventTime[12];
  char newEventTime[12];

  strcpy(newEventTime, strcat(start, end));

  if (curEvent != NULL) {
    strcpy(curEventTime, curEvent->startTime);
    strcat(curEventTime, curEvent->endTime);

  }

  if (curEvent == NULL) {
    headEvent->next = newEvent;
  }

  else if (strcmp(newEventTime, curEventTime) < 0) {
    newEvent->next = curEvent;
    headEvent->next = newEvent;
  }

  else {
```

```
    if (curEvent->next != NULL) {
      strcpy(curEventTime, curEvent->next->startTime);
      strcat(curEventTime, curEvent->next->endTime);
      while (curEvent->next && strcmp(newEventTime, curEventTime) > 0) {
        curEvent = curEvent->next;
        if(curEvent->next){
            strcpy(curEventTime, curEvent->next->startTime);
            strcat(curEventTime, curEvent->next->endTime);
        }
      }

        if(strcmp(newEventTime, curEventTime) == 0){
            if(favorite>curEvent->next->favorite){
                newEvent->next = curEvent->next;
                curEvent->next = newEvent;

            }
            else{
                while (curEvent->next && (strcmp(newEventTime,
curEventTime) == 0) && favorite <= curEvent->next->favorite) {
                    curEvent = curEvent->next;

                    if(curEvent->next){
                        strcpy(curEventTime, curEvent->next->startTime);
                        strcat(curEventTime, curEvent->next->endTime);
                    }
                   }
                    newEvent->next = curEvent->next;
                    curEvent->next = newEvent;
                }

        }
        else{
            newEvent->next = curEvent->next;
            curEvent->next = newEvent;
        }
    }
    else{
        newEvent->next = curEvent->next;
        curEvent->next = newEvent;
    }
  }
}

// get all event that outdated and insert from function stackHelper
void insertToStack(Stack* outdated,Year* headYear){
```

```
    Year *curYear = headYear->next;
    while (curYear) {
      Month *curMonth = curYear->month->next;
      while (curMonth) {
        Day *curDay = curMonth->day->next;
        while (curDay) {
          Event *curEvent = curDay->event->next;
          while (curEvent) {
            //Check if in stack
            char tmpEvent[3000],startEvent[12],endEvent[12];
            int favoriteEvent = curEvent->favorite;
            strcpy(tmpEvent,curEvent->event);
            strcpy(startEvent,curEvent->startTime);
            strcpy(endEvent,curEvent->endTime);
            favoriteEvent = curEvent->favorite;

stackHelper(outdated,curYear->year,curMonth->month,curDay->day,startEven
t,endEvent,tmpEvent,favoriteEvent);
            curEvent = curEvent->next;
          }
          curDay = curDay->next;
        }
        curMonth = curMonth->next;
      }
      curYear = curYear->next;
    }
}
```

ฟังก์ชัน stackHelper ทำหน้าที่ช่วยให้ insert Task ลงใน Stack ได้อย่างถูกตำแหน่ง โดยรับข้อมูลของ Task ทั้งหมด แล้วเรียกใช้ฟังก์ชัน searchByYearFromOutdated เพื่อเช็กว่า Task ที่จะ insert มี Struct year หรือยัง ถ้ายังไม่มีให้เรียกใช้ฟังก์ชัน createYear เพื่อสร้าง Struct year ใหม่ จากนั้นเช็กเดือน, วัน และเวลาของ Task ว่าต้องสร้าง Struct ใหม่หรือต้องหาตำแหน่งที่ควรอยู่ใหม่

insertToStack ()

```
// get all event that outdated and insert from function stackHelper
void insertToStack(Stack* outdated,Year* headYear){
    Year *curYear = headYear->next;
    while (curYear) {
      Month *curMonth = curYear->month->next;
      while (curMonth) {
        Day *curDay = curMonth->day->next;
        while (curDay) {
          Event *curEvent = curDay->event->next;
          while (curEvent) {
            //Check if in stack
            char tmpEvent[3000],startEvent[12],endEvent[12];
            int favoriteEvent = curEvent->favorite;
            strcpy(tmpEvent,curEvent->event);
            strcpy(startEvent,curEvent->startTime);
            strcpy(endEvent,curEvent->endTime);
            favoriteEvent = curEvent->favorite;

stackHelper(outdated,curYear->year,curMonth->month,curDay->day,startEven
t,endEvent,tmpEvent,favoriteEvent);
            curEvent = curEvent->next;
          }
          curDay = curDay->next;
        }
        curMonth = curMonth->next;
      }
      curYear = curYear->next;
    }


}
```

ฟังก์ชัน insertToStack ทำหน้าที่ insert Task ทั้งหมดลงใน Stack โดยไล่นำข้อมูลของแต่ละ Task มาเรียกใช้ฟังก์ชัน stackHelper เพื่อนำข้อมูลลง Stack

cleanOutdatedData ()

```
// Remove Outdated data form tree and push into out dated stack
void cleanOutdatedData(Year* headYear,Stack* stack){

   Year *curYear = headYear->next;
    displayCurrentTime();
    if (curYear == NULL) {
        //printf("No task\n");
    }
    else {

    while (curYear) {
      Year* reuseYearnext = curYear->next;
      int status = 0;
      Year* newHeadList = createYear(-1);
      unsigned int stackCheck = 0;
      unsigned int cntDel = 0;
      Month *curMonth = curYear->month->next;
      while (curMonth) {
        Day *curDay = curMonth->day->next;
        while (curDay) {
          Event *curEvent = curDay->event->next;
          while (curEvent) {
            unsigned int check = 0;

            check =
isOutdatedData(curYear->year,curMonth->month,curDay->day,curEvent->endTi
me);

            if(check == 1){
                stackCheck=1;
                //pop node from priorityQueue then create temp list for
pushing on stack.
                char tmpStartTime[12];
                char tmpEndTime[12];
                char tmpEvent[3000];
                strcpy(tmpStartTime,curEvent->startTime);
                strcpy(tmpEndTime,curEvent->endTime);
                strcpy(tmpEvent,curEvent->event);

insert(newHeadList,curYear->year,curMonth->month,curDay->day,tmpStartTim
e,tmpEndTime,tmpEvent,0);

                cntDel++;
```

```
            }
            curEvent = curEvent->next;
          }
          int prevDay = curDay->day;
          curDay = curDay->next;
          if(cntDel>0){

            deleteFirstEvent(headYear,cntDel, curYear->year,
curMonth->month, prevDay, &status);

            cntDel=0;
          }
        }
        curMonth = curMonth->next;
      }

      if(stackCheck == 1){
        insertToStack(stack, newHeadList);
      }
      else{
        free(newHeadList);
      }
      //Year has been free ;-;
      if(status == 1){
         if(reuseYearnext != NULL)curYear = reuseYearnext;
         else curYear = NULL;
      }
      else
            curYear = curYear->next;
    }
  }
}
```

ฟังก์ชัน cleanOutdatedData ทำหน้าที่ย้าย Task ที่เลยกำหนดเวลามาแล้วไปลงใน Stack โดยการไล่ Tree แล้วเรียกใช้ฟังก์ชัน isOutdatedData ในการเช็กว่า Task ที่เลยกำหนดเวลามาแล้วในวันเดือนปีนั้นมีกี่ Task จากนั้นก็สร้าง Tree ขึ้นมาใหม่แล้วเรียกใช้ฟังก์ชัน insert เพื่อนำข้อมูลของ Task ที่เลยกำหนดเวลามาแล้วใส่ลงใน Tree ใหม่ ต่อด้วยการลบ Task ที่เลยกำหนดเวลามาแล้วทิ้งด้วยฟังก์ชัน deleteFirstEvent หลังจากไล่ Tree ได้ครบทุกเดือนแล้ว ให้เรียกใช้

ฟังก์ชัน insertToStack ถ้ามี Task ในปีนั้นๆ ที่เลยกำหนดเวลา เพื่อที่จะนำ Task ที่เลยกำหนดเวลาใส่ลงใน Stack

saveToText ()

```
// Save to Text File
void saveToText(Year *headYear) {
  FILE *fp = fopen("dateList.txt", "w");
  Year *curYear = headYear->next;
  if (curYear == NULL) {
    printf("No task\n");
  } else {
    while (curYear) {
      Month *curMonth = curYear->month->next;
      while (curMonth) {
        Day *curDay = curMonth->day->next;
        while (curDay) {
          Event *curEvent = curDay->event->next;
          while (curEvent) {
            fprintf(fp, "%d %d %d %s %s %s %d\n", curYear->year,
                    curMonth->month, curDay->day, curEvent->event,
                    curEvent->startTime, curEvent->endTime,
curEvent->favorite);
            curEvent = curEvent->next;
          }
          curDay = curDay->next;
        }
        curMonth = curMonth->next;
      }
      curYear = curYear->next;
    }
  }
  fclose(fp);
}
```

ฟังก์ชัน saveToText ทำหน้าที่ save Task ทั้งหมดลงในไฟล์ .txt โดยการไล่ Tree แล้วนำข้อมูลของ Task print ลงในไฟล์ .txt ในรูปแบบ “ปี เดือน วัน ชื่อ เวลาเริ่ม เวลาสิ้นสุด ความสำคัญ”

readTextFile ()

```
// Read Text File
void readTextFile(Year *headYear) {
  FILE *fp = fopen("dateList.txt", "r");
  int year, month, day, favorite;
  char start[12], end[12], buffer[64], event[32];
  // char *event;
  while (fgets(buffer, 64, fp) != NULL) {
    if (buffer) {
      sscanf(buffer, "%d %d %d %s %s %s %d\n", &year, &month, &day,
event,
             start, end, &favorite);
      insert(headYear, year, month, day, start, end, event, favorite);
    } else
      break;
  }

  fclose(fp);
}
```

ฟังก์ชัน readTextFile ทำหน้าที่แปลงไฟล์ .txt ออกมาเป็นข้อมูลของ Task แล้วนำไปเรียกใช้ฟังก์ชัน insert เพื่อที่จะสร้าง Tree

saveToOutdated ()

```
// Save for each year in stack
void saveToOutdated(Stack *outdated) {
  int prevTop = outdated->top;
  while (outdated->top != -1) {
    Year *outdatedHeadYear = peek(outdated);
    outDatedList(outdatedHeadYear, outdatedHeadYear->next->year);
    pop(outdated);
  }
  outdated->top = prevTop;
}
```

ฟังก์ชัน saveToOutdated ทำหน้าที่ save Task ที่เลยกำหนดเวลามาแล้วทั้งหมดลงในไฟล์ .txt โดยไล่ Stack เรียกใช้ฟังก์ชัน peek เพื่อนำข้อมูลของ Task มาเรียกใช้ฟังก์ชั่น outDatedList

outDatedList ()

```
// Save to outDated.txt file
void outDatedList(Year *headYear, int year) {
  Year *prevYear = headYear;
  Year *curYear = searchByYear(headYear, &prevYear, year);

  FILE *fp = fopen("dateList.txt", "a");
  if (curYear) {
    Month *curMonth = curYear->month->next;
    while (curMonth) {
      Day *curDay = curMonth->day->next;
      while (curDay) {
        Event *curEvent = curDay->event->next;
        while (curEvent) {
          fprintf(fp, "%d %d %d %s %s %s %d\n", curYear->year,
curMonth->month,
                  curDay->day, curEvent->event, curEvent->startTime,
                  curEvent->endTime, 0);
          curEvent = curEvent->next;
        }
        curDay = curDay->next;
      }
      curMonth = curMonth->next;
    }
  }
  fclose(fp);
}
```

ฟังก์ชัน outDatedList ทำหน้าที่ผนวก Task ที่เลยกำหนดเวลามาแล้วทั้งหมดลงในไฟล์ .txt โดยการไล่ Tree แล้วนำข้อมูลของ Task print ลงในไฟล์ .txt ในรูปแบบ “ปี เดือน วัน ชื่อ เวลาเริ่ม เวลาสิ้นสุด ความสำคัญ”

int main ()

```c
#include "event.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
int main() {
  Year *Patitin = createYear(-1);
  Stack *outdate = createStack(100);
  char buffer[10];
  int cmd;
  int year = -1, month = -1, day = -1, favorite = -1;
  char start[12], end[12], event[3000];
  // DailyNotification
  displayCurrentDateTask(Patitin);
  readTextFile(Patitin);

  while (1) {

    // Clear Outdated
    cleanOutdatedData(Patitin, outdate);

    // Main function
    printf("----------------------\n");
    printf("1  - Create task\n");
    printf("2  - Display all task\n");
    printf("3  - Display Outdated\n");
    printf("4  - Display Favorited\n");
    printf("5  - Delete task\n");
    printf("6  - Search\n");
    printf("7  - Edit\n");
    printf("99 - Exit\n");
    printf("Insert command :");

    fgets(buffer, 10, stdin);
    enterRemover(buffer);
    while (sscanf(buffer, "%d", &cmd) != 1) {
      printf("wrong command format please try again\n");
      printf("Inset command :");
      fgets(buffer, 10, stdin);
      enterRemover(buffer);
    }

    if (cmd == 1) {
      userInput(&year, &month, &day, start, end, event, &favorite, 1);
      insert(Patitin, year, month, day, start, end, event, favorite);
    }

    else if (cmd == 2) {
      int allTask = 0;
```

```
    displayAll(Patitin, 2, &allTask);
  }

  else if (cmd == 3) {
    displayOutdated(outdate);
  }

  else if (cmd == 4) {
    displayFavorite(Patitin);
  }

  else if (cmd == 5) {
    printf("Please input date to show all Lists in that day\n");
    userInput(&year, &month, &day, start, end, event, &favorite, 2);
    deleteEvent(Patitin, outdate, year, month, day);
  }

  else if (cmd == 6) {
    printf("Please insert date and name to search\n");
    printf("(leave \"--\" if don't want to search in that fields)\n");
    printf("*** for now you can only search by only one fields at the time "
           "***\n");
    userInput(&year, &month, &day, start, end, event, &favorite, 3);

    // searchByDate
    if (day != -1 && month != -1 && year != -1 && !strcmp(event, "--")) {
      displayByDate(Patitin, year, month, day);
    }
    // searchByDay
    else if (day != -1 && month == -1 && year == -1 && !strcmp(event, "--")) {
      displayByDay(Patitin, day);
    }
    // searchByMonth
    else if (day == -1 && month != -1 && year == -1 && !strcmp(event, "--")) {
      displayByMonth(Patitin, month);
    }
    // searchByYear
    else if (day == -1 && month == -1 && year != -1 && !strcmp(event, "--")) {
      displayByYear(Patitin, year);
    }
    // searchByName
    else if (day == -1 && month == -1 && year == -1 && strcmp(event, "--")) {
      displayByEvent(Patitin, event);
    }
  }

  else if (cmd == 7) {
    printf("Please input date to show all Lists in that day\n");
    userInput(&year, &month, &day, start, end, event, &favorite, 2);
    edit(Patitin, year, month, day);
  }
```

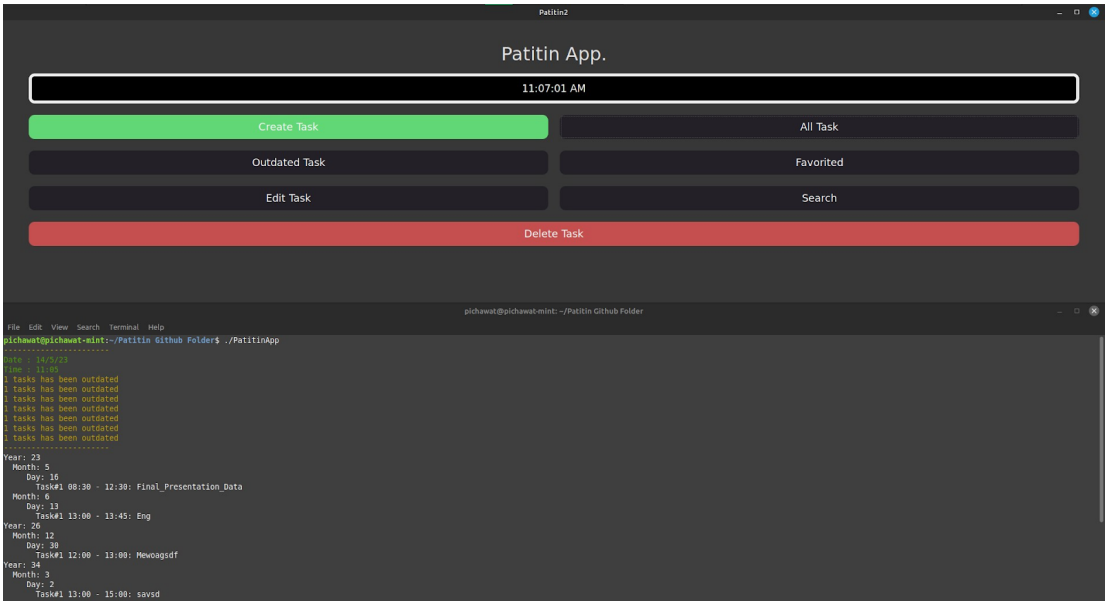
```
    else if (cmd == 99)
      return 0;
    saveToText(Patitin);
    saveToOutdated(outdate);
  }

  return 0;
}
```

ฟังก์ชัน main จะคอยรับตัวเลขจากผู้ใช้งานเพื่อเรียกการใช้งาน function แต่ล่ะ function ตามที่กำหนดไว้

วิธีการใช้งาน

เมื่อเริ่มต้นใช้งาน โปรแกรมจะแสดงวันที่และเวลา ณ ปัจจุบันออกมาและหากมีไฟล์ .txt ที่เก็บข้อมูลและในข้อมูลนั้นมี task ณ วันที่นั้น โปรแกรมจะแสดง Task ที่อยู่ในวันนั้นทั้งหมดออกมา ผู้ใช้งานสามารถกด function หน้า UI เพื่อใช้งานแต่ล่ะ fucntion



ภาพตัวอย่าง UI ของ โปรแกรม

แสดงการทำงานของแต่ละฟังก์ชัน

1. Create task

ทำหน้าที่สร้าง Task ใหม่ โดยรับวัน เดือน ปี เวลา ชื่อ และความสำคัญของ Task ที่ต้องการสร้างจาก User พร้อมกับเช็กความถูกต้องของข้อมูล และสร้างเป็น Task ใหม่เพื่อนำไปจัดเก็บใน Priority Queue ซึ่งจัดเรียงจากวันและเวลาของ Task โดยเรียงจากน้อยไปมาก

```
Insert command :1
input dd/mm/yy (01/01/23): 01/01/23
input time to start and end in format (xx:yy)
Time to start: 09:00
Time to end: 12:00
Name your Task: Task1
Please rate how important this task is
inset "-1" if you think that not important
input( any number that more than or equal zero ) : -1
```

ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Create Task

2. Display all task

ทำหน้าที่แสดงผล Task ที่ยังไม่เลยกำหนดเวลาทั้งหมด โดยแสดงผล Task ตามวันและเวลาที่ใกล้กับเวลาปัจจุบันมากที่สุดก่อน

```
Insert command :2
----------------------
Year: 23
  Month: 5
     Day: 15
       Task#1 09:00 - 12:00: Task5
     Day: 16
       Task#1 08:00 - 11:00: Task6
     Day: 17
       Task#1 09:00 - 12:00: Task7
     Day: 18
       Task#1 08:00 - 11:00: Task8
```

ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Display all task

3. Display outdated

ทำหน้าที่แสดงผล Task ที่เลยกำหนดเวลาไปแล้วทั้งหมด โดยแสดงผล Task ตามวันและเวลาที่ไกลจากเวลาปัจจุบันมากที่สุดก่อน

```
Insert command :3
Year: 23
Month 1
Day 1:
   Task#1 09:00 - 12:00: Task1
Day 2:
   Task#1 08:00 - 11:00: Task2
Day 3:
   Task#1 09:00 - 12:00: Task3
Day 4:
   Task#1 08:00 - 11:00: Task4
```

ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Display outdated

4. Display favorited

ทำหน้าที่แสดงผล Task ที่มีความสำคัญและยังไม่เลยกำหนดเวลาทั้งหมด (Task ที่เลยกำหนดเวลาเราจะถือว่าไม่มีความสำคัญ) โดยแสดงผล Task ตามวันและเวลาที่ใกล้กับเวลาปัจจุบันมากที่สุดก่อน

```
Insert command :4
Year: 23 Month: 5  Day: 17 Task: Task7 Time: 09:00 - 12:00 Favorite: 5
Year: 23 Month: 5  Day: 18 Task: Task8 Time: 08:00 - 11:00 Favorite: 1
```

ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Display favorited

5. Delete task

ทำหน้าที่ลบ Task ที่ต้องการ โดยรับวัน เดือน และปีของ Task ที่ต้องการลบจาก User พร้อมกับเช็กความถูกต้องของข้อมูล โปรแกรมจะแสดงผล Task ที่มีวัน เดือน และปีตรงกับข้อมูลจาก User โดยที่แต่ละ Task จะมีหมายเลขกำกับ จากนั้นจะให้ User เลือก Task ที่ต้องการลบจากหมายเลขกำกับ

```
Insert command :5
Please input date to show all Lists in that day
input dd/mm/yy (01/01/23): 01/01/23
--------------OUTDATED!!!--------------
Year: 23 Month: 1 Day: 1
   Task#1 09:00 - 12:00: Task1
Select Task that you want to delete by number: 1
Remember delete can't be undo
Choice 1: Yes (Delete it)
Choice 2: No (Cancel it)
Select choice : 1
Deletion Outdated Success!!!
```

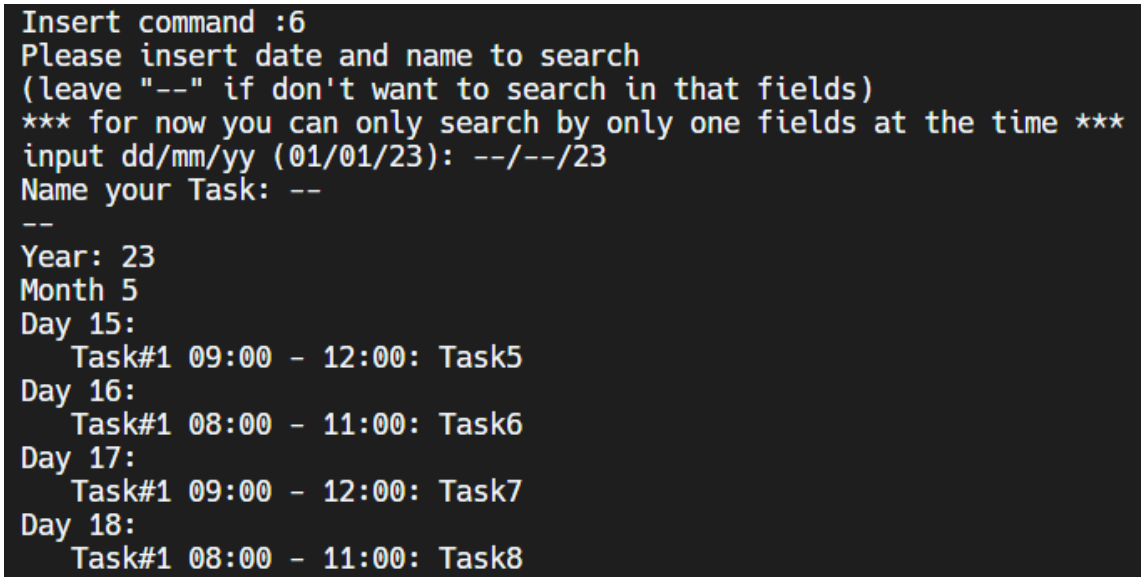
ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Delete task โดยที่ Task นั้นเลยกำหนดเวลาไปแล้ว

```
Insert command :5
Please input date to show all Lists in that day
input dd/mm/yy (01/01/23): 15/05/23
Year: 23 Month: 5 Day: 15
   Task#1 09:00 - 12:00: Task5
Select Task that you want to delete by number: 1
Remember delete can't be undo
Choice 1: Yes (Delete it)
Choice 2: No (Cancel it)
Select choice : 1
Deletion Success!!!
```

ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Delete task โดยที่ Task นั้นยังไม่เลยกำหนดเวลา

6. Search

ทำหน้าที่ค้นหา Task ที่ยังไม่เลยกำหนดเวลาจากวันหรือชื่อและแสดงผล โดยรับวัน เดือน ปี หรือชื่อเพียงอย่างใดอย่างหนึ่งของ Task ที่ต้องการค้นหาจาก User พร้อมกับเช็กความถูกต้องของข้อมูล และแสดงผล Task ที่มีวัน เดือน ปี หรือชื่อตรงกับข้อมูลจาก User โดยแสดงผล Task ตามวันและเวลาที่ใกล้กับเวลาปัจจุบันมากที่สุดก่อน



ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Search

7. Edit

ทำหน้าที่แก้ไข Task ที่ยังไม่เลยกำหนดเวลา โดยรับวัน เดือน และปีของ Task ที่ต้องการแก้ไขจาก User พร้อมกับเช็กความถูกต้องของข้อมูล โปรแกรมจะแสดงผล Task ที่มีวัน เดือน และปีตรงกับข้อมูลจาก User โดยที่แต่ละ Task จะมีหมายเลขกำกับ จากนั้นจะให้ User เลือก Task ที่ต้องการแก้ไขจากหมายเลขกำกับ และแก้ไขข้อมูลของ Task ที่เลือก โดยใส่ข้อมูลใหม่ลงไป

```
Insert command :7
Please input date to show all Lists in that day
input dd/mm/yy (01/01/23): 15/05/23
Year: 23 Month: 5 Day: 15
   Task#1 09:00 - 12:00: Task5
Select Task that you want to edit by number: 1
Please insert all data you want to edit
(leave "--" if don't want to edit in that fields we will use the old ones)
input dd/mm/yy (01/01/23): 19/--/--
input time to start and end in format (xx:yy)
Time to start: 10:00
Time to end: 13:00
Name your Task: Task9
Task9
Please rate how important this task is
input( any number that more than or equal zero ) : -1
```

ภาพแสดงตัวอย่างการทำงานของฟังก์ชัน Edit

```
Insert command :2
----------------------
Year: 23
  Month: 5
     Day: 16
       Task#1 08:00 - 11:00: Task6
     Day: 17
       Task#1 09:00 - 12:00: Task7
     Day: 18
       Task#1 08:00 - 11:00: Task8
     Day: 19
       Task#1 10:00 - 13:00: Task9
```

ภาพแสดงผลลัพธ์ของฟังก์ชัน Edit

8. Exit : ทำหน้าที่จบการทำงานของโปรแกรม

สรุปผล

จากการใช้งานโปรแกรม PatiTin_2 โปรแกรมสามารถทำงานได้ปกติไม่มีข้อผิดพลาดใดๆ และใช้โครงสร้าง Priority queue , Stack, Liked list, Tree และ Array เพื่อนำไปใช้ในการเขียนในแต่ละ function ของโปรแกรมได้ทำงานได้ตามที่กำหนด อย่างไรก็ตามหลังจากการทดสอบการใช้งานโปรแกรมพบว่ามีข้อจำกัดของโปรแกรมดังนี้

- ในการแสดงผลแต่ละ task หากมีข้อมูลจำนวนมากอาจยากต่อการตรวจสอบ
- ในการค้นหา task สามารถค้นหาได้เพียงครั้งละ 1 ประเภทเท่านั้น
- หาก user กรอกข้อมูลผิด โปรแกรมจะไม่ได้บอกถึงว่า user กรอกข้อมูลผิดแบบใดเพียงแค่บอกว่าผิดแล้วให้กรอกใหม่เพียงเท่านั้น
- โปรแกรมไม่สามารถยกเลิกการกระทำที่ทำไปแล้ว เช่น หากเลือก Insert แล้วต้องการยกเลิกขณะที่กรอกข้อมูลจะไม่สามารถทำได้
- ในการกรอกปีของโปรแกรมจำเป็นต้องกรอกเพียงเลขสองหลักเท่านั้น (20XX)